An Emission and Discard Priority Scheme for Optical Burst Switched Networks

Pascal E.K. Acquaah

Supervisor: H. Anthony Chan

Submitted to the Department of Electrical Engineering

in partial fulfillment of the requirements for the degree

of Master of Science in Electrical Engineering

at the

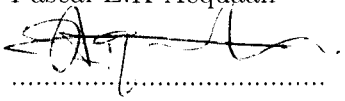
University of Cape Town

August 31, 2007

DECLARATION

I declare that the thesis, **An Emission and Discard Priority Scheme for Optical Burst Switched Network**, is my own work, that it has not been submitted for any degree or examination in any other university, and that all the sources I have used or quoted have been indicated and acknowledged by references.

Full name: Pascal E.K Acquaah Date: August 31, 2007

Signed: 

ACKNOWLEDGMENTS

I would like to thank the following individuals who have provided me with invaluable help in completing this project:

- The Lord My GOD for his constant miracles in my life.
- My mother for her moral and financial support to help me get to this stage in my life.
- Ben G. Muwonge for his constant support and intellectual assistance
- All the members of the communications and research group for their valuable comments.
- Prof H. Anthony Chan for advice in completing this project.

ABSTRACT

Optical burst switching (OBS) is a promising technology designed to meet the growing demands for internet bandwidth and better Quality of Service (QoS). This technology provides all optical and high speed switching to overcome the bottleneck of electronic routers in the core network. In this thesis, I describe several critical issues that affect OBS networks. I highlight the need to resolve contention efficiently and cost-effectively to improve QoS in OBS networks.

Techniques to resolve contention include wavelength conversion, deflection routing, optical buffering, and burst segmentation. Amongst these techniques, deflection routing is the most cost-effective because it requires no extra hardware and offers high throughput at low loads. However, deflection routing has shortcomings such as high burst loss at high loads and a high number of late packet arrivals, which hinder its performance. Hence, the aim of our study is to reduce the number of late packet arrivals due to burst deflections and to increase the efficiency of deflection routing. For this purpose, we propose an emission and discard priority (EDP) scheme.

The EDP scheme assigns emission and discard priorities to bursts based on their QoS requirements. This type of burst differentiation allows routers to deflect bursts selectively and efficiently in the core network. Through simulations, I compare the performance of the deflection with EDP scheme, the no-deflection (drop policy) scheme, and the deflection scheme. I show that the deflection with EDP scheme has a higher throughput than the no-deflection scheme, and a higher throughput than the deflection scheme for loads $L \geq 0.7$. Furthermore, I show that the deflection with EDP scheme has a lower proportion of late packet arrivals than the deflection scheme at all loads. As a result, the deflection with the EDP scheme outperforms the deflection scheme in terms of goodput for $L \geq 0.7$. Although the deflection scheme has higher goodput than the deflection with EDP scheme at low loads ($L < 0.7$), the

deflection with EDP scheme performs fewer burst deflections, with a minimal loss in goodput. Hence, the deflection with EDP scheme has a higher goodput per deflection ratio, and is therefore more efficient than the deflection scheme without EDP.

Contents

List of Figures

List of Tables

List of Abbreviations

ATM	Asynchronous Transfer Mode
E-O	Electrical –Optical conversion
FDL	Fiber Delay Line
Gbps	Giga Bits per Second
IP	Internet Protocol
JET	Just-Enough-Time
JIT	Just-In-Time
LAN	Local Area Network
LRD	Long Range Dependence
Mbps	Mega Bits per Second
OBS	Optical Burst Switching
O-E	Optical-Electrical conversion
O-O-O	All Optical
OXC	Optical Cross-Connect
QoS	Quality of Service
QoE	Quality of Experience
SONET	Synchronous Optical Network
Tbps	Terabit Bits per Second
TCP	Transmission Control Protocol

combines the properties of both optical circuit switching and optical packet switching (Figure 1.2).

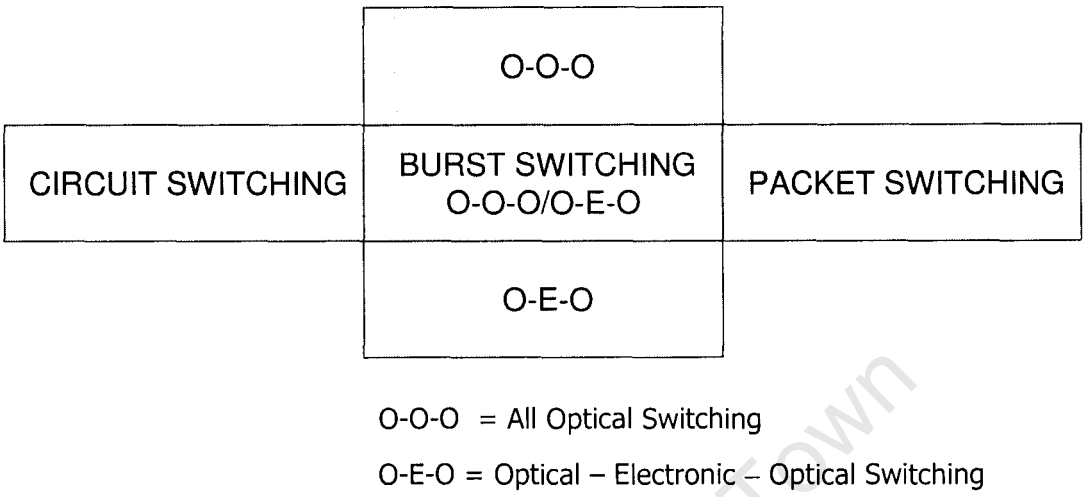


Figure 1.2 Merging Switching Technologies.

In **Optical Burst Switching**, incoming packets from various sources are assembled into blocks of data called bursts, at the edge nodes of the network. When a burst is generated, it is buffered in a queue for a waiting time referred to as *offset time* (Figure 1.3). During the offset time, a control packet, which is referred to as burst header packet (BHP), is sent ahead of the burst to configure switches along the burst's path (Figure 1.3). When the offset period expires, the burst follows the predefined route set by its BHP and is transmitted through the network all-optically (Figure 1.3). As a result, no electronic or optical buffering is necessary for the burst. However, because the BHP is processed electronically, the offset time should be long enough to prevent the data burst from catching up with the BHP.

During its electronic processing, the BHP also specifies the duration of the incoming burst to allow the node to reconfigure its switch for other incoming bursts. Hence, a higher degree of statistical multiplexing is achieved because multiple traffic sources can access the same set of resources. Furthermore, due to data being transmitted in large bursts, the switching requirements of OBS are less than the switching requirements of OPS.

1 Introduction

1.1 Optical Switching

The rapid growth of the internet has stimulated a pressing need for higher bandwidth. Optical networks, with their high transmission capacity, have the potential to meet the increasing demands for internet bandwidth. To take advantage of this high transmission capacity, an optical technology called Dense Wavelength Division Multiplexing (DWDM) was introduced. This technology exploits the high potential bandwidth (over 50 Tb/s) of the optical fiber by dividing it into different channels using a different wavelength of light for each channel. In this way, different types of data traffic from different technologies such as IP and SONET/SDH can be multiplexed onto a single fiber line. DWDM technology achieves over 160 wavelengths/channels, with each wavelength carrying up to 10 Gbps. Wavelength capacities of 40 Gbps are expected in the near future.

Various switching technologies have been proposed to support the transport of data over DWDM networks. **Optical Circuit Switching** (OCS) was the first switching technology to enable the transmission of data over DWDM networks (Figure 1.1). In Optical Circuit Switching, an end-to-end connection (lightpath) is set up between a node pair using a dedicated wavelength on every link along the path. The time needed to set up a lightpath is on the order of milliseconds, and therefore to be efficient the data transmission time should be on the order of minutes. Long data transmission times make OCS ideal for voice traffic. However, the burstiness of data traffic [1, 2] requires short data transmission times. Short data transmission times lead to the inefficient utilization of bandwidth. OCS is therefore not convenient for the transport of data traffic. The burstiness of data traffic calls for the development of other methods for the transport of data over DWDM networks.

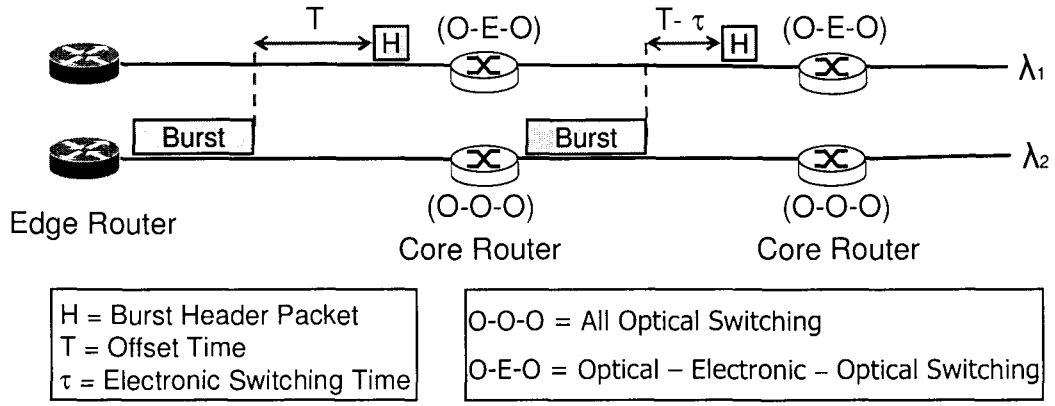


Figure 1.3 Transmission of a Burst Header Packet and a Data Burst on Separate Channels.

Table 1-1 summarizes the available optical switching technologies. We can see that OBS appears to have the best of both the OCS and OPS paradigms, while avoiding their pitfalls.

Table 1-1 Comparison of Optical Switching Technologies.

Optical Switching Technology	Bandwidth Utilization	Connection Setup	Optical Processing	Processing Overhead	Data Traffic Support
OCS	Low	High	Not Required	Low	Low
OPS	High	Low	Required	High	High
OBS	**High**	**Low**	**Not Required**	**Low**	**High**

1.2 An Optical Burst Switched Network for Africa

Although the demand for internet bandwidth is growing around the world, it is limited on the African continent. This limited demand of internet bandwidth is mainly due to a poor telecommunications infrastructure, which isolates Africa from the rest of the world. The poor state of telecommunications in Africa is further aggravated by the high cost of telephone calls to other African countries and the loss

of telecommunication revenue to Europe. The internet is an unlimited source of information that is necessary to boost the development of Africa. The current penetration of internet usage in Africa represents only 1.1% of its population. To provide internet to African countries in a cost-effective manner, it is necessary to develop an efficient fiber-optic backbone that interconnects Africa with the rest of the world [3].

Optical burst switching (OBS) has been identified as a cost-effective and promising switching paradigm for transferring data over the optical fiber. The rapid progress of the OBS concept from theoretical investigations to actual implementations [4] makes it a viable candidate to support the transfer of data over the African network in the future.

1.3 Problem Statement - Improving QoS in Optical Burst Switched Networks

The inherent challenge in OBS networks is to provision QoS in a simple and effective way. The objective of OBS is to eliminate the bottleneck that electronic routers cause in the core of optical networks. To eliminate this bottleneck, it is important to minimize the complexity of the core network by pushing the network intelligence to the edge. In this way, we can reduce the delays incurred in transmitting data. However, OBS compromises the guarantee of packet transmissions, through wavelength path reservation (optical circuit switching), with bandwidth utilization. This compromise means that OBS uses a one-way reservation scheme (optical packet switching), which maximises link utilization and minimizes transmission delay. As a result, the level of QoS drops because the probability of contention between bursts increases in the network. Resolving contention in an efficient and cost-effective manner therefore becomes a high priority to improve QoS in the OBS domain.

Techniques to resolve contention include wavelength conversion, optical buffering, deflection routing, and burst segmentation. The focus of this thesis is on

deflection routing. Studies have shown that deflection routing increases throughput in OBS network [5]. The main advantage of deflection routing is that it requires no extra hardware in the core network and is therefore cost-effective. However, deflection routing has a negative impact on burst loss at high traffic loads. In addition, deflection routing increases the average end-to-end transmission delay of bursts and therefore leads to late packet arrivals. As a result, the amount of data that reaches destination within its delay requirements is reduced. This amount of data is also referred to as goodput.

In this thesis, I introduce an emission and discard priority (EDP) scheme for OBS. This EDP scheme consists in providing efficient service class differentiation at the edge and the core of the network based on traffic requirements. I implement the EDP scheme to improve the efficiency of deflection routing and reduce its end-to-end transmission delays. In this way, the effect of deflection routing on late packet arrivals is reduced and the QoS of the OBS network is enhanced in terms of goodput. Eliminating the major shortcomings of deflection routing will help to establish it as an effective contention resolution scheme for OBS networks.

1.4 Research Objectives

The objectives of this study are as follows:

- Investigate the impact of resolving contention with deflection routing on OBS networks.
- Propose and simulate an emission and discard priority (EDP) scheme to improve the efficiency of deflection routing.
- Improve the goodput of deflection routing in OBS networks with the EDP scheme.
- Determine whether deflection routing is an effective contention resolution scheme for OBS networks.

1.5 Scope and Limitations of Research

This study does not investigate the other shortcomings of deflection routing which include out-of-order packet arrivals and the need to set an offset time that caters for extra burst delays inside the network. These shortcomings of burst deflections are considered to have secondary impact in proportion to late packet arrivals in the network.

1.6 Thesis Outline

This section describes the outline of this thesis.

In Chapter 2, we provide a detailed review of the current literature in specific areas of OBS. We describe fundamental issues such as burst assembly, signaling, scheduling, contention resolution and QoS.

Chapter 3 highlights specific issues concerning contention resolution techniques in OBS networks. We introduce an emission and discard priority (EDP) scheme to eliminate the shortcomings of deflection routing, and improve QoS of OBS networks.

In Chapter 4, we explain the details of our experiment. We describe the details of our simulation environment as well as the parameters used to simulate the proposed deflection routing with EDP scheme.

Chapter 5 evaluates the performance of the deflection with EDP scheme based on QoS metrics such as throughput, delay and goodput. Our simulation results compare the cases of the no-deflection scheme, the deflection scheme and the deflection with EDP scheme.

In Chapter 6, we summarize the findings of our experiment and give recommendations for future work.

2 Optical Burst Switched Networks

The objective of this chapter is to survey previous work done in Optical Burst Switching (OBS). The chapter outline is as follows. Section 2.1 introduces the concept of OBS and its proposed network architecture. In Section 2.2 and Section 2.3, we review methods of burst assembly and describe the reservation techniques used in OBS. Section 2.4 evaluates proposed scheduling schemes for OBS. Section 2.5 discusses different contention resolution techniques for the core network, and Section 2.6 describes several schemes used to achieve QoS in an OBS network.

2.1 Optical Burst Switching Technology and Architecture

The concept of Optical Burst Switching (OBS) was proposed in [6-8]. The motivation behind the OBS concept is a network that supports the burstiness of data traffic and requires limited or no delay at the intermediate nodes in the network. OBS aims at transmitting data transparently over DWDM links at high speed and with high efficiency to provide good quality of service (QoS) to all users. In this section, we define the OBS network architecture and the technology needed to implement it. Figure 2.1 presents the network architecture of an OBS network.

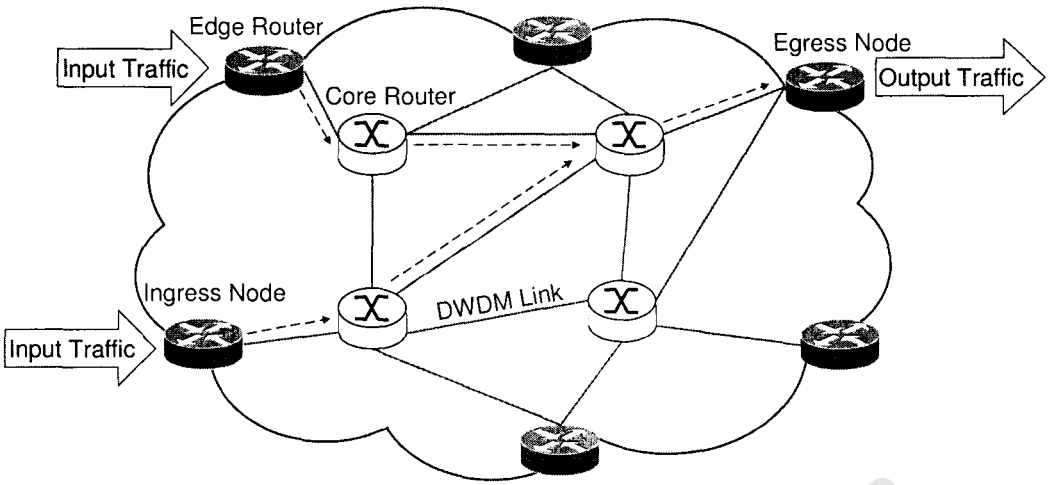


Figure 2.1 An Optical Burst Switched Network.

An OBS network consists of edge routers and core routers that are interconnected using DWDM links (Figure 2.1). In Optical Burst Switching, packets are transported throughout the network as blocks of data called bursts. A burst consists of multiple IP packets aggregated at the ingress of an OBS network. Edge routers are the transition point between an OBS core network and other access networks. They are responsible for assembling packets into bursts and scheduling them for transmission on outgoing wavelengths. Core routers are responsible for forwarding data bursts inside the core network (Figure 2.1).

The ingress edge node assembles incoming packets with the same destination (egress edge node) and/or quality of service parameters (e.g. delay requirements) into a burst. This process is called burst assembly or burstification (Figure 2.2).

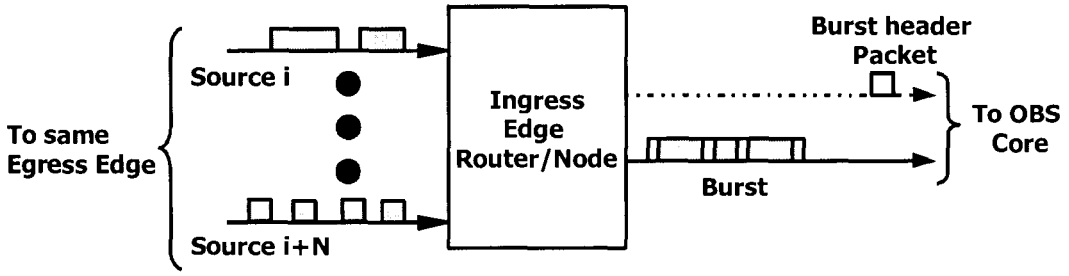


Figure 2.2 Burst Assembly at the edge of an OBS Network.

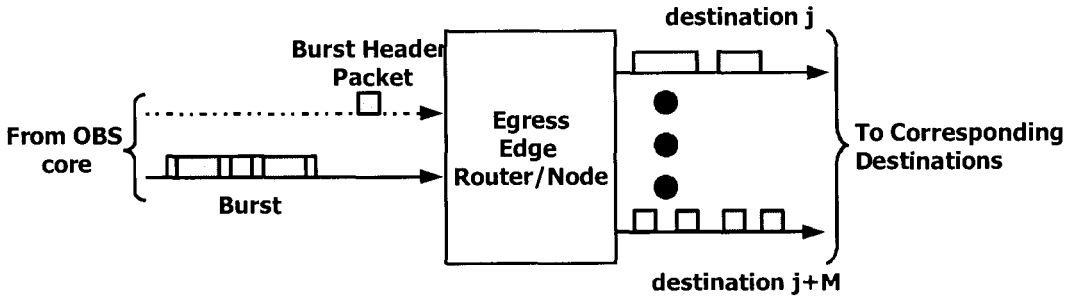


Figure 2.3 Burst Disassembly at the edge of an OBS Network.

Burst disassembly or deburstification is the reverse process, and occurs at the egress edge node (Figure 2.3). During deburstification, the egress node disassembles bursts back into individual packets and forwards them to their respective destinations.

Figure 2.4 reveals the different functionalities implemented within an OBS network. The ingress edge node deals with burst assembly, routing and wavelength assignment, and scheduling. The core node does the signalling, scheduling within the core and resolves contention issues within the network. The egress edge node disassembles bursts and forwards packets to their respective destinations.

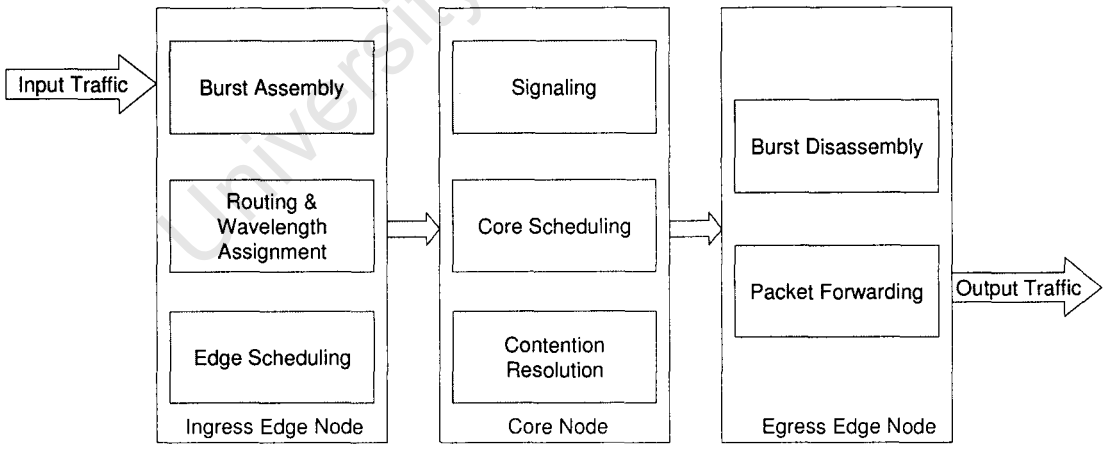


Figure 2.4 OBS Functional Diagram.

Generally, a node encompasses the functionalities of both an edge and a core node. The core node consists of a switch control unit (SCU) and an optical cross-

connect (OXC) (Figure 2.5). The SCU interprets header packets, schedules bursts, resolves contention, maintains a forwarding table, controls the switching matrix, rewrites header packets, and changes output wavelengths when necessary. When a BHP arrives at the core node, the SCU identifies the output port and consults the routing signalling processor to locate the output port. If the output port is available, the SCU configures the OXC to switch the data burst when it arrives. In case the output port is not available, the SCU applies the contention policy of the network to resolve the problem.

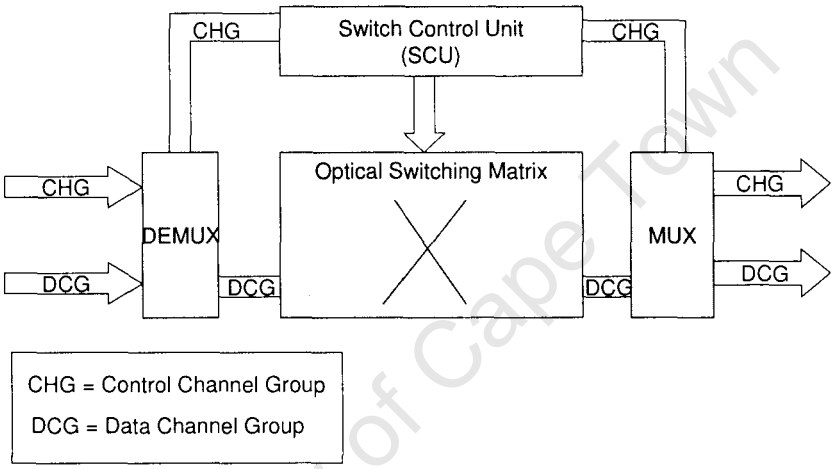


Figure 2.5 Core Router Architecture.

The edge node consists of a switching matrix and burst assembly units (Figure 2.6). The switching matrix forwards the packets that arrive at the egress node to the burst assembler. At this point, the burst assembler assembles packets depending on their destination and QoS parameters. Each assembly queue corresponds to a specific egress edge router and a specific class. When bursts reach the egress edge router, they are disassembled into packets and forwarded to the higher layers of the network.

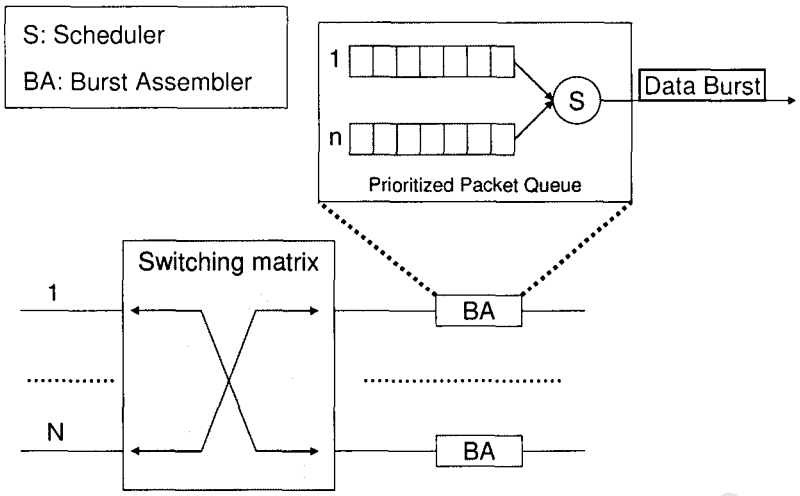


Figure 2.6 Architecture of an OBS Ingress Edge Router.

When designing an OBS network, physical-layer issues such as attenuation, dispersion, and fiber nonlinearities have to be considered. Although these issues are common to all-optical networks, they may cause serious concern in the design and implementation of an OBS networks [9]. Refer to [9, 10]for further details on the physical constraints of building OBS networks.

In the next section, we describe and analyse the current ways of assembling packets at the ingress node of an OBS network.

2.2 Burst Assembly Techniques

Burst assembly is the process of aggregating packets from different sources into bursts at the ingress node of an OBS network. The edge node buffers packets electronically depending on their class and their destination (egress edge node). The key factors in burst assembly are when to create bursts and when to send them into the core network. These factors are important because of the bufferless nature of an OBS network. Hence, the way in which packets are assembled has a direct influence on the performance of the network in terms of bandwidth efficiency, burst loss rate and transmission delay.

To assemble packets in an efficient manner, researchers have proposed a number of burst assembly techniques. These assembly techniques can be classified as timer-based, burst length-based or both [11-14].

2.2.1 Timer-based Burst Assembly Algorithm

Algorithm I: Fixed Assembly Period Algorithm [14]

This algorithm assembles packets based on a fixed time interval T. Packets that arrive during T are assembled into a burst (Figure 2.7). The timer is set when the first packet arrives at T_0. At T_1, a time out occurs, and all the packets that arrived during T form a burst. Although this algorithm is simple, the assembly time interval T is fixed. Thus, if T is too large, the packet delay can be intolerable, and packets can arrive late at their destination (egress edge node); if T is too small, a large number of short bursts is generated, which causes a high control overhead at the core routers within the network. The timer interval needs to be carefully set to achieve an acceptable performance in the core network. We note that the Fixed Assembly Period algorithm fails to take into account the burstiness and the QoS requirements of incoming data packets.

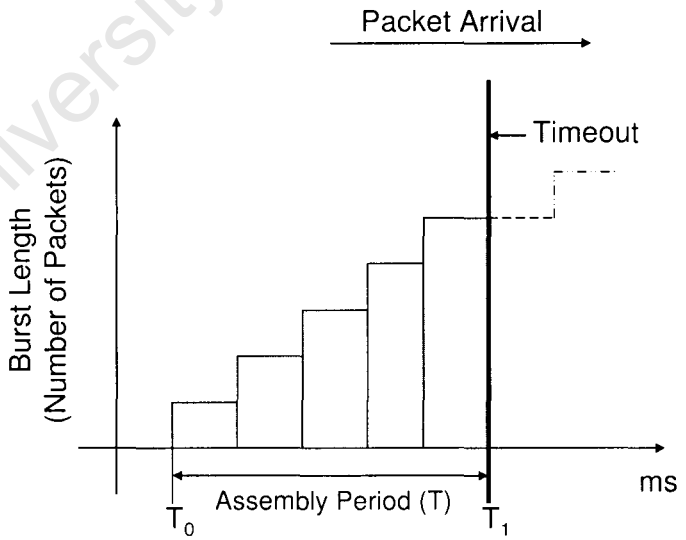


Figure 2.7 Fixed Timer-based Assembly.

2.2.2 Burst Length-based Assembly Algorithm

Algorithm II: Maximum Burst Length-based Assembly

In this algorithm, bursts are assembled using a maximum burst size. The assembly process is similar to the time based assembly algorithm. The only difference is that packets are assembled using a fixed burst size (Figure 2.8); hence, when the sum in bytes of the packets reaches the maximum burst size, a burst is generated. This mechanism ensures that a sufficient number of packets are assembled at once, and thus prevents a high control overhead at the core routers (due to short bursts). However, the end-to-end delay that packets will experience is not guaranteed, which implies that packets can arrive late at destination. QoS, in terms of end-to-end delay requirements, is therefore an important factor to consider in the burst assembly process.

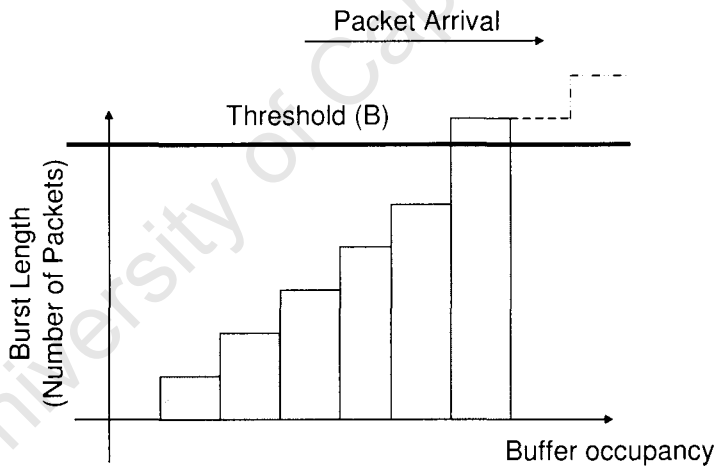


Figure 2.8 Burst Length-based Assembly.

2.2.3 Mixed Timer/Burst Length-based Assembly Algorithms

A variety of mixed timer/ burst length based assembly algorithms were proposed to overcome the deficiencies associated with timer based and burst length based assembly algorithms. The general idea is that packets can be sent into the networks when a timer (T) expires or when the burst reaches a specific threshold (B) as shown in Figure 2.9.

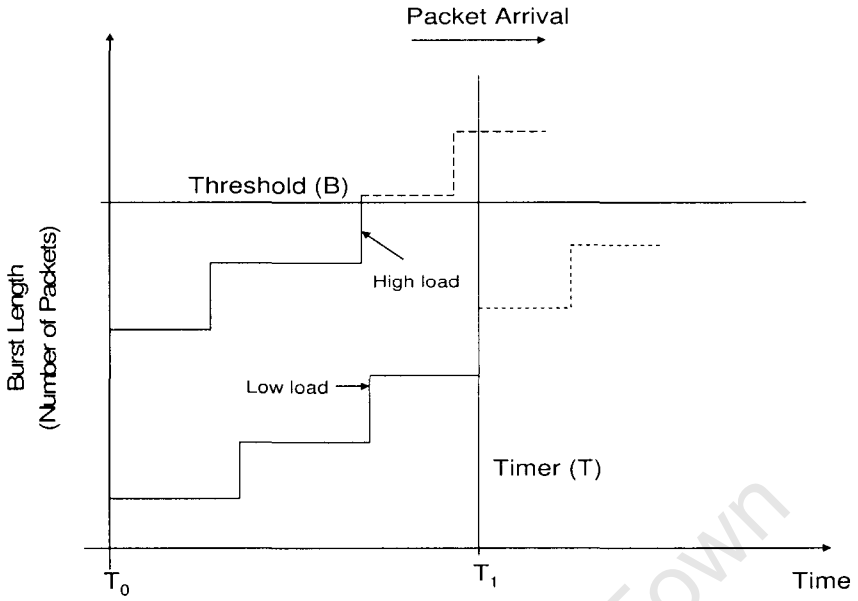


Figure 2.9 Mixed timer/burst length based assembly

Algorithm III: Minimum Burst Length Maximum Assembly Period Algorithm [14]

This assembly algorithm is based on a minimum burst length and a maximum period of assembly to generate bursts. A burst is created whenever the minimum burst length is exceeded or a timeout occurs, whichever comes first. The minimum burst length ensures decent burst sizes in the network while the maximum assembly period guarantees that the first packet in the burst does not miss its deadline. The minimum burst length is set to be lower than the average burst length and the maximum assembly period is approximately the difference between the retransmit time-out value and the round trip time value of the packet (in the case of TCP traffic). Setting the value of the minimum burst length to be lower than the average burst length hinders the throughput performance of the network because the burst length is not optimal. Furthermore, input packets in a burst have various delay requirements and thus, the first packet in the burst does not necessarily have the most critical deadline. Thus, the maximum assembly period may not be appropriate for the packets that are assembled after the first packet in the burst. In other words, the maximum assembly period does not guarantee that subsequent packets in the

burst will meet their end-to-end delay requirements. We note that this algorithm is designed for TCP traffic. [14] concluded that algorithm III performs similarly to algorithm I for TCP traffic.

Algorithm IV: Max-Time-Min-Max-Length Burst assembly [12]

The Max-Time-Min-Max-Length burst assembly algorithm is based on a maximum assembly period, a minimum burst length and a maximum burst length. The assembly period is fixed just like algorithms I and III. A minimum burst length ensures a limit on the control overhead in the network similarly to algorithm III except that bursts are not sent out when the minimum burst length is exceeded. Instead, bursts are sent out only (1) when the timeout expires or (2) when the maximum burst length is reached. A burst is sent out (3) if and only if its size/length is greater than or equal to the minimum burst length. In the case where condition (1) is satisfied but condition (3) is not met, the size of the data burst is increased with padding and sent out immediately. We note that the assembly period can be set to ensure that the burst sizes/lengths never reach the maximum burst length, in which case the assembly period is the primary criteria in the process of burst assembly.

Algorithm V: Non periodic time interval burst assembly [11]

This algorithm is different from all the previous algorithms because it uses a non-periodic time interval to assemble bursts. In other words, the assembly period varies every time a new burst is assembled. Packets are assembled based on their destination, a maximum assembly time (which is non-periodic) and a maximum burst length. For simplicity, we assume that all the incoming packets have the same destination. A timer is started when a packet arrives at the edge node. When a time interval T is reached or the size of the burst reaches the maximum burst length M, the burst is assembled with a length L.

The timer is then reset until the next packet arrives. This assembly algorithm assembles different traffic classes in different queues to account for QoS in the OBS

network. Hence, this algorithm can enhance the burst loss rate and the end-to-end packet delay of an OBS network provided an appropriate T is selected whenever a burst is assembled. However, how to select T is still an open question [11].

Other adaptive assembly algorithms were proposed to dynamically adjust the time or the burst length parameters based on real time traffic measurements. They provide a better performance but require a high operational complexity [15].

2.2.4 The Effect of Burst Assembly on OBS networks

Although burst assembly creates a delay at the edge of the network, studies have shown that burst assembly can have a *smoothing* effect on bursty data traffic [14]. This effect implies that an appropriate burst assembly mechanism can reduce the variance in the number of bursts/packets that arrive at a node simultaneously, and the variance of the data rate. Thus, burst assembly can improve the overall performance of the network, in terms of data loss rate and throughput. The smoothing effect of burst assembly on data traffic is only over a short range (short time scale). Although [13] claimed that burst assembly could reduce the long range (large to infinite time scale) dependence property of input data traffic, [11, 12] showed that the long range dependency of data traffic remains unchanged even after burst assembly. On the other hand, [16] showed that the long-range dependency of data traffic does not affect the performance of the OBS network, in terms of burst loss rate, due to its bufferless nature. Because the OBS core network is bufferless, the burst assembly process at the edge has a major impact on the performance of the network. Thus, a higher emphasis needs to be put on the edge rather than the core of the network.

In the analysis of the packet loss rate in any network, factors such as the inter-arrival rate distribution and the packet length distribution of packets need to be considered. Although it seems logical to consider the inter-arrival time distribution of bursts when measuring the performance of an OBS network, studies have shown that the inter-arrival time distribution of packets have a negligible effect on the core network's performance [17]. This conclusion was based on the assumption

of exponentially distributed burst lengths. On the other hand, [11] concluded that because burst lengths follow a Gaussian distribution over constant time interval, it may not be appropriate to use an exponential burst length distribution for the performance measurements of an OBS core network. The effect of the variance in the inter-arrival time distribution of bursts therefore remains an open issue.

[17] showed that the burst length distribution and its average value, which are both dependent on the burst assembly period, affect the performance of an OBS network. These parameters affect the performance of the core network, in terms of the end-to-end packet delay and the burst drop rate in the OBS network. If the burst assembly period is too long, bursts can arrive late at destination. On the other hand, if the burst assembly period is too short, a large number of short bursts are sent in the network, which causes a higher load of bursts at the core routers [15]. Thus, the period of assembly should be optimal for best results.

Resource reservation is the next step following burst assembly. In the next section, we describe how resources are allocated for the transmission of bursts. We classify signaling schemes and give an overview of the early burst transmission/reservation protocols. We then describe and compare the major OBS signalling schemes.

2.3 Signaling Schemes

In OBS, resources need to be allocated to ensure the smooth transition of bursts over an optical core of the network. Signaling schemes, also known as burst reservation protocols, are implemented to allocate the required resources to configure the optical switches for incoming bursts.

Burst reservation protocols are classified with the following characteristics:

- *One-way reservation* (no acknowledgment), *two-way reservation* (acknowledgment), or *hybrid reservation* (partial acknowledgment).
- *Source-initiated, destination-initiated, or intermediate-node-initiated.*

- *Persistent* (waits for blocked resources to be released) or *non-persistent* reservation (uses contention resolution mechanism when resources are blocked).
- *Immediate reservation* or *delayed reservation.*
- *Explicit* (separate control message) or *implicit* release of resources.
- *Centralized signaling* or *distributed signaling.*

Burst reservation protocols in OBS are derived from early bursts transmission protocols such as tell-and-wait (TAW) and tell-and-go (TAG). These protocols were proposed in [18] for ATM networks. In TAW, when a burst is ready for transmission, a request is sent from the source to the destination to reserve bandwidth resources for the burst. When each intermediate node along the way receives the request, it reserves an output port for the burst. If the request reaches its destination after reserving the necessary resources at all the links along the path, an ACK packet is sent back to inform the source to send out the burst immediately. If resource reservations are not successful, a NAK is returned to release the previously reserved resources for the burst. The source resends a request after a backoff period. We classify TAW as a two-way reservation protocol.

In the TAG, the node transmits bursts without making any reservations in contrast to the TAW protocol. Bursts need to be delayed at each intermediate node to allow time for the reservation of an output port. Fiber Delay Lines (FDLs) are used to provide a fixed delay at each input port for the incoming bursts as in Terabit Burst Switching [19]. If an output port is not available, reservation fails and a NAK packet is sent back to the source. In this way, the source can initiate the retransmission of bursts after a backoff time. We classify TAG as a one-way reservation protocol.

Just-In-Time (JIT) [8] and the Just-enough-Time (JET) [6, 7] are the most prevailing reservation protocols in optical burst switching. They are both one-way reservation protocols, similar to tell-and-go (TAG), and do not require any kind of optical buffering at each intermediate node. They accomplish this by allowing the

control packet to carry an offset time (Figure 1.3). At each intermediate node, the offset time between the control packet and its corresponding burst reduces to account for the processing delay of the control packet.

2.3.1 Immediate Reservation (Just-In-Time)

The Just-In-Time (JIT) reservation scheme is an example of an immediate reservation mechanism. At an intermediate node, an output port is reserved immediately after the arrival and processing of the control packet (BHP). If no output port is free when the BHP arrives, it is rejected and the corresponding burst is dropped. Figure 2.10 illustrates the reservation mechanism of JIT on a single channel.

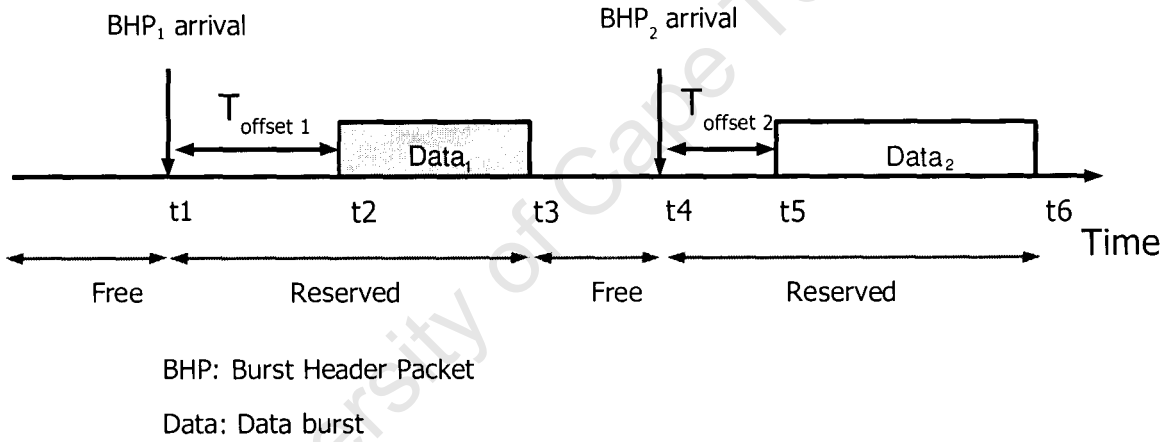


Figure 2.10 An Immediate Reservation Mechanism (JIT).

The channel can be either free or reserved. When the first burst header packet (BHP_1) arrives at the node, the channel is free, and therefore available for reservation. After accepting BHP_1, the node reserves the channel for a period t3-t1 for the incoming data burst ($Data_1$). The time interval t2-t1 is the offset time between the BHP and the data burst. Any BHP that arrives between t1 and t3 is rejected since the channel is reserved until the last bit of the data burst ($Data_1$) is transmitted. The length of the period t3-t1 is equal to the sum of the offset time ($T_{offset1}$) and the length of the burst ($Data_1$). We note that the next scheduled burst

($Data_2$) on the channel is not necessarily the next burst to arrive at the node but rather the next burst to arrive at the node when the wavelength is free (after t3).

2.3.2 Delayed Reservation (Just-Enough-Time and Horizon)

JET [6, 7] and Horizon [19] both use a delayed reservation mechanism. Here, an output wavelength/channel is reserved just before the arrival of the first bit of the burst at the intermediate node. If no output channel is available, the burst control packet (BHP) is rejected and the corresponding burst is dropped. Because bursts do not arrive at the node one right after the other, intervals called voids are created on the channels. The main difference between JET and Horizon is that JET attempts to fill these voids whereas Horizons does not.

Delayed reservation scheme without void filling (Horizon)

Under Horizon, the node keeps track of the scheduling horizon of each channel in the fiber. The scheduling horizon of a channel is the latest reservation time of bursts on a channel (i.e. the time after which no bursts are reserved on the channel). In order to reserve a channel for an incoming burst, channels for which the scheduling horizon is earlier than the burst arrival time are considered. Amongst these channels, the one with the latest scheduling horizon is reserved for the incoming burst. In other words, a reservation can be made on a channel if and only if the burst arrival time is later than the scheduling horizon of the channel [20]. If no output channel is available when the control packet (BHP) arrives, it is rejected and the corresponding burst is dropped. Although Horizon attempts to minimize the void size between successive bursts, it does not maintain any information about the size of the voids. Thus, Horizon cannot fit any short bursts in those voids, leading to a waste of resources. Figure 2.11 illustrates the delayed reservation mechanism of Horizon on a single channel.

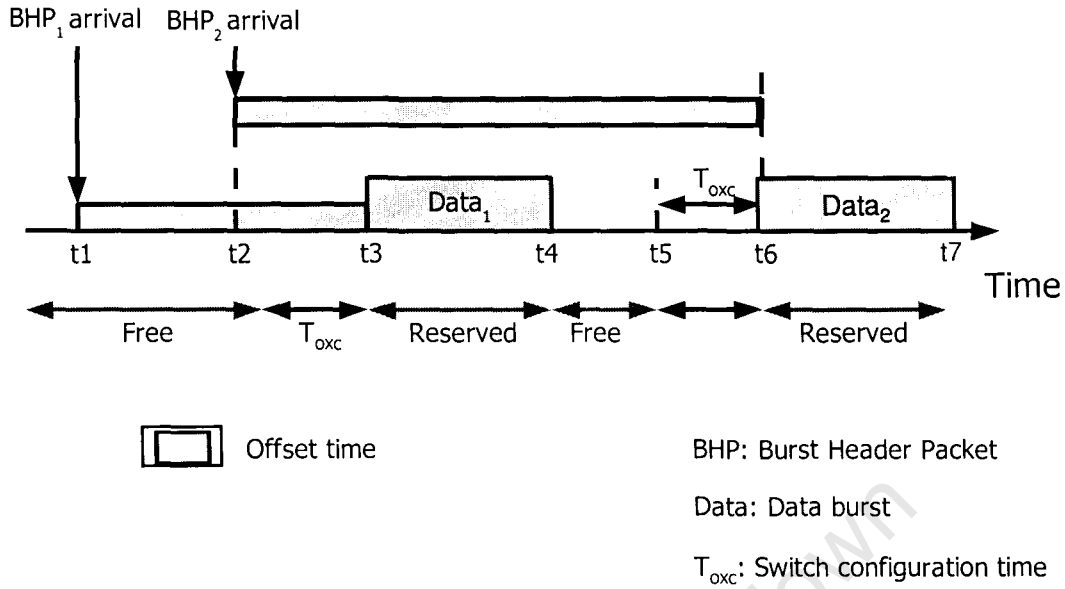


Figure 2.11 A Delayed Reservation Mechanism without Void-filling (Horizon).

As shown in Figure 2.11, the node does not reserve the channel immediately after accepting BHP_1 in contrast to the JIT reservation protocol. Instead, the node schedules $Data_1$ for transmission between t3 and t4. t4 becomes the scheduling horizon of the channel. The channel remains free until a short period (T_{oxc}) before the arrival of the $Data_1$. The node needs a time T_{oxc} to configure the switch for the incoming burst ($Data_1$). No other bursts can be scheduled during T_{oxc}, so the channel is essentially reserved for T_{oxc} + Reserved. After switching $Data_1$, the node frees the channel. It then reconfigures the switch at t5 for the incoming data burst ($Data_2$) at t6. The delayed reservation of $Data_1$ allows $Data_2$ to be scheduled when BHP_2 arrives at t2. At this point, t7 becomes the scheduling horizon of the channel. The channel is released at t7 after switching $Data_2$, allowing other bursts to be scheduled.

Note that the offset of a burst ($Data_2$) may overlap with the offset and/or transmission of another ($Data_1$). The order of arrival of BHPs determines the reservation order of bursts (i.e. bursts are reserved on a channel in a First Come First Serve manner) [20].

Delayed reservation scheme with void filling (JET)

JET is the most prevalent reservation protocol with void filling. Studies in [21] have shown that JET outperforms the Horizon and JIT reservation mechanisms in terms of burst loss rate. Under JET, an output channel is reserved for bursts whose arrival time is either (1) later than the scheduling horizon of the channel or (2) coincides with a void on the channel (2). If the arrival time of a burst corresponds to a void, *the end of that burst must be earlier than the end of the corresponding void* to successfully schedule the burst. If no output channel is available when the control packet (BHP) arrives, it is rejected and the corresponding burst is dropped. Figure 2.12 illustrates the delayed reservation mechanism with void filling on a single channel. We note that bursts scheduled based on condition (2) would be rejected under the Horizon reservation protocol.

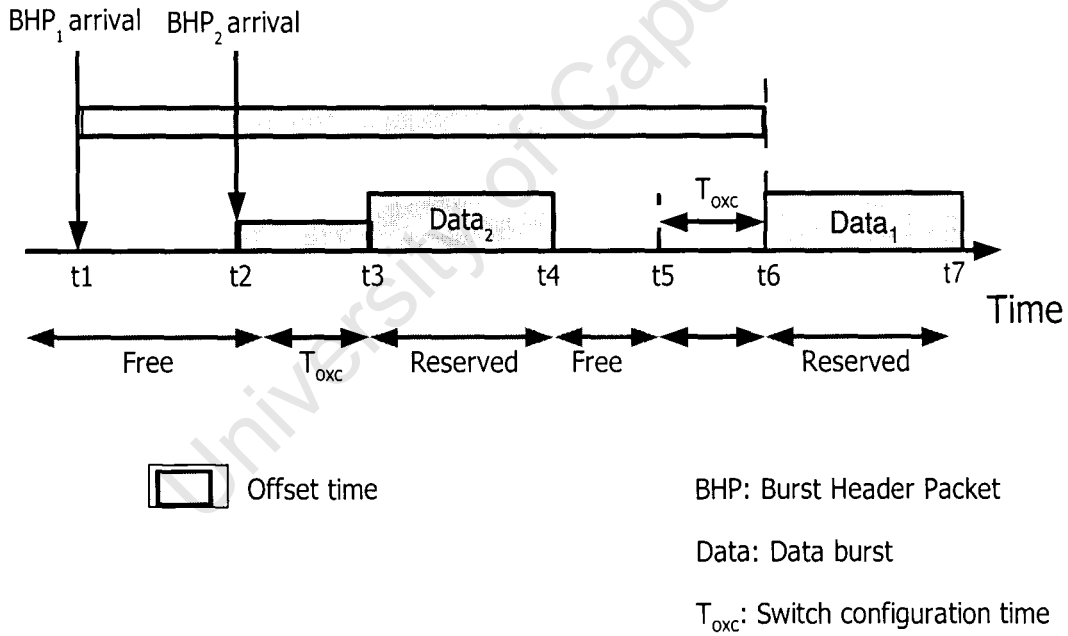


Figure 2.12 A Delayed Reservation with Void-filling (JET).

JET protocol maintains the start and end times of all the scheduled bursts. This information allows the node to keep track of all the voids on every channel. Thus, JET is the most complex of all the OBS reservation protocols because it has

to maintain more channel information. It also outperforms both JIT and Horizon in terms of burst loss rate. Figure 2.12 illustrates the void-filling characteristic of the JET protocol. Although BHP_2 arrives at the node after BHP_1, the node is able to schedule $Data_2$ at t3. This is because the arrival (t3) and departure (t4) times of $Data_2$ are earlier than the arrival time (t6) of $Data_1$. Thus, reservations are not necessarily made in a FCFS manner under the JET protocol. Under Horizon, BHP_2 would be rejected and $Data_2$ dropped.

2.3.3 Evaluation of Signalling Schemes

Table 2-1 shows proposed signalling schemes including the early signalling protocols, TAW and TAG. All signalling schemes use a one-way reservation scheme except TAW, which uses a two-way reservation scheme. The main benefit of using one-way reservation is the lower end-to-end delay needed to transmit bursts across the optical core at the cost of a high loss of burst due to the contention of resources in the network (Table 2-1). We observe that the direction of the signalling schemes is the main factor when considering their impact on delay and loss. The other factors have a negligible effect on delay and loss.

Table 2-1 Summary of Burst Reservation Schemes.

Signalling	**Direction**	**Initiation**	**Reservation**	**Release**	**Delay**	**Loss**	**Void-Filling**
TAW	Two-way	Source/ destination	Explicit	Explicit	High	Low	No
TAG	One-way	Source	Implicit	Implicit	Low	High	No
JIT	One-way	Source	Explicit	Explicit	Low	High	No
JET	One-way	Source	Implicit	Implicit	Low	High	Yes
Horizon	One-way	Source	Implicit	Implicit	Low	High	No

Another important performance parameter to consider is bandwidth utilization. The choice of a signalling scheme has a significant effect on the efficient usage of bandwidth in the network. For example, the JIT protocol is simpler than the JET reservation protocol. It does not require any complex void filling techniques unlike JET. It is therefore easier to implement in hardware with our current technology. Although the simplicity of JIT makes it easier to implement, it has a reduced efficiency in terms of bandwidth utilization. This is because it does not take advantage of the voids on the channels as in JET and does not attempt to minimize the voids on channels as in Horizon. Furthermore [21] showed that the efficiency of JIT reduces significantly as the offset increases in comparison to the JET and Horizon reservation protocols.

The point of OBS is to provide a transition from optical circuit switching (OCS) to optical packet switching (OPS). The transition from OBS to OPS will require the complexity of current technologies to increase. The implementation of optical header processing and optical buffering will require a higher degree of complexity than the implementation of JET or Horizon. This complexity is the cost of developing technology. Furthermore, the need to improve the network efficiency in terms of bandwidth utilization validates the added complexity of JET and Horizon. Implementing the best reservation schemes (JET) can only be a step towards the implementation of a seamless optical network. We therefore assume that our OBS network uses JET for resource reservation.

In OBS, bursts sent out in the network are buffered for limited periods because of the lack of optical RAM (optical buffering). Therefore, scheduling bursts in a timely manner is a major concern. To avoid high loss, bursts need to be scheduled in a fast and bandwidth efficient manner at the intermediate nodes. In the next section, we describe and compare current OBS scheduling techniques.

2.4 Scheduling Schemes

Scheduling algorithms are closely related to the hardware architecture of the OBS network. To implement scheduling techniques in an OBS network, a core router

needs to have wavelength conversion capability. The use of Tunable Wavelength Converters allows the burst scheduler to schedule a burst from its input port to any output port. Below, we describe various scheduling algorithms.

2.4.1 LAUC / Horizon

Amongst all the scheduling algorithms, First Fit (FF) is the simplest and most intuitive. However, this algorithm is not efficient because it searches for the first available channel in the link and schedules the burst. [11] proposed the LAUC (Latest Available Unscheduled Channel) algorithm, which is similar to the Horizon scheduling/reservation protocol that Turner designed in [19]. In LAUC/Horizon, a scheduler keeps track of the latest scheduling horizon of each channel on the link. An incoming burst is scheduled on the channel with the latest scheduling horizon provided it is still earlier than the arrival time of the burst. LAUC/Horizon aims at minimizing voids between channel horizons and burst arrival times. In Figure 2.13, the LAUC/Horizon algorithm selects channel C_1.

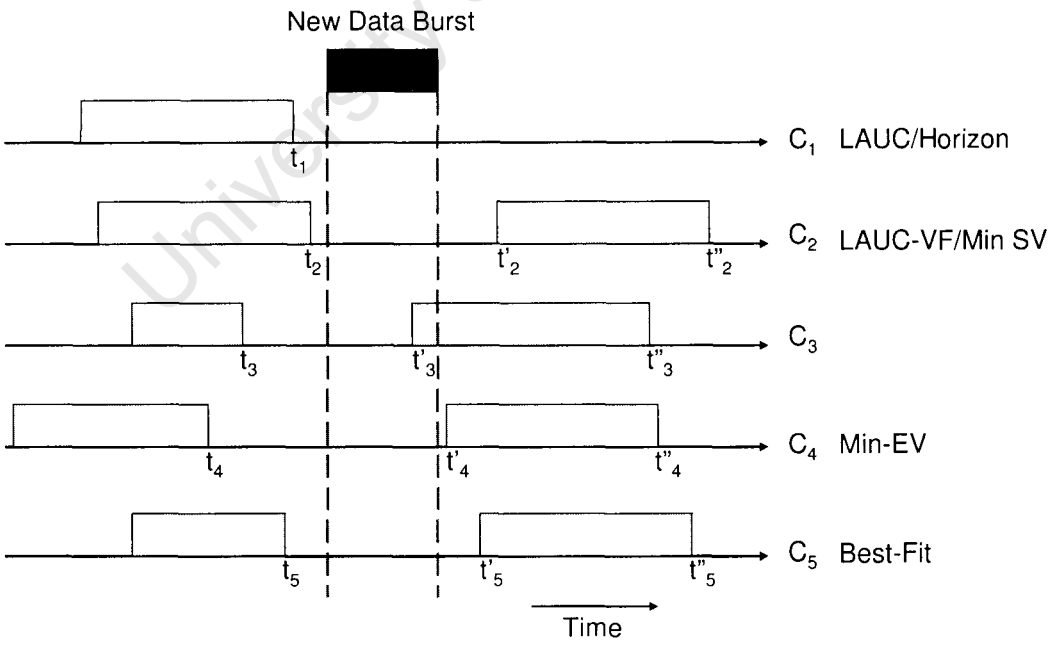


Figure 2.13 Illustration of OBS Scheduling Algorithms.

2.4.2 LAUC-VF

The main advantage of LAUC and Horizon is their relative simplicity and good performance in terms of execution time. However, they waste bandwidth resources because they cannot schedule bursts in voids between existing reservations as seen in Figure 2.13 (t2'-t2 void on channel C_2). LAUC-VF (LAUC with void-filling) was proposed in [11] as an improvement to LAUC. LAUC-VF aims at making reservations within existing voids. Thus, LAUC-VF schedules the burst on channel C_2 rather than channel C_1 (Figure 2.13). In this way, LAUC-VF makes better use of the available bandwidth and leaves opportunities for future bursts to be scheduled on channel C_1. The drawback of LAUC-VF is its high runtime complexity.

2.4.3 Min-SV, Min-EV and Best-Fit

While LAUC/Horizon trade bandwidth efficiency for fast running time, LAUC-VF trades fast running time for bandwidth efficiency. Min-SV (minimum starting void), Min -EV (minimum ending void) and Best-Fit [22], were therefore proposed to achieve the running time of LAUC/Horizon while maintaining the high bandwidth efficiency of LAUC-VF.

Min-SV, Min-EV and Best-Fit are all variants of LAUC-VF. Although Min-SV is functionally the same as LAUC-VF (Figure 2.13), it achieves a lower running time complexity using a technique from computational geometry [15]. Min-EV on the other hand, minimizes the distance between the end of a new reservation and the end of an existing reservation (Figure 2.13). Min-SV and Min-EV are therefore conceptually symmetric to each other [22] (Channel C2 and C4 in Figure 2.13).

The Best-Fit scheduling algorithm minimizes the total length of the starting and ending voids when scheduling a burst between two existing reservations (Figure 2.13). Table 2-2 shows the differences between the scheduling algorithms, which are described in terms of runtime complexity, and bandwidth utilization.

We use the following notations in Table 2-2 [15] to compare the different scheduling algorithms:

- W: Number of wavelengths at each output port.
- M: Maximum number of reservations on all channels.
- Horizon $_i$: Horizon of the i^{th} data channel.
- $S_{i,j}$ and $E_{i,j}$: Starting and ending time of the j^{th} reservation on channel i.

Table 2-2 Comparison of OBS Scheduling Algorithms.

Scheduling algorithm	**Time complexity**	**State information**	**Bandwidth utilization**
LAUC/Horizon	O(W)	Horizon $_i$	Low
LAUC-VF	O(W log M)	$S_{i,j}$ $E_{i,j}$	High
Min-SV / Min-EV	O(log M)	$S_{i,j}$ $E_{i,j}$	High
Best-Fit	$O(\log^2 M)$	$S_{i,j}$ $E_{i,j}$	High

As seen in Table 2-2 LAUC/Horizon has the lowest bandwidth utilization. It also has the best runtime complexity, which increases linearly with the number of wavelengths on each output port. The other scheduling techniques have high bandwidth utilization but differ in their runtime complexity. Min-SV and Min-EV algorithms run faster than LAUC-VF.

Although Min-SV and Min-EV are conceptually symmetric to each other in the way they schedule bursts, their performances differ. While the loss rate of Min-SV is about 20% lower than that of Min-EV, Min-EV has runs several times faster than Min-SV [22]. No experimental results were obtained with the Best-Fit algorithm

in [22] but the analysis suggests high bandwidth utilization and a runtime that is faster than LAUC-VF but slower than Min-SV and Min-EV (Table 2-2).

Although Min-SV and Min-EV perform better than other void filling algorithms (Table 2-2), we prefer the use of LAUC-VF for simulation purposes. We note that although the use of better scheduling algorithms (e.g. Min-SV) would improve our experimental results; they will not be a factor in the analysis of our results. Thus, we choose to use LAUC-VF for scheduling bursts in our OBS network.

In OBS, the use of one-way reservation protocols like JET means that bursts are sent into the network without acknowledgment. However, the lack of acknowledgment means that intermediate nodes have to resolve the potential contention of bursts for resources. In the next section, we describe how contention is resolved in an OBS network.

2.5 Contention Resolution Techniques

Contention occurs when multiple bursts compete for the same wavelength/channel on the same output port simultaneously. In this thesis, we refer to the burst that arrives first at the node as the *original burst* and the burst that follows as the *contending burst*. Burst scheduling and contention resolution are linked closely because an efficient burst scheduling algorithm can help reduce contention. Figure 2.14 illustrates current contention resolution options.

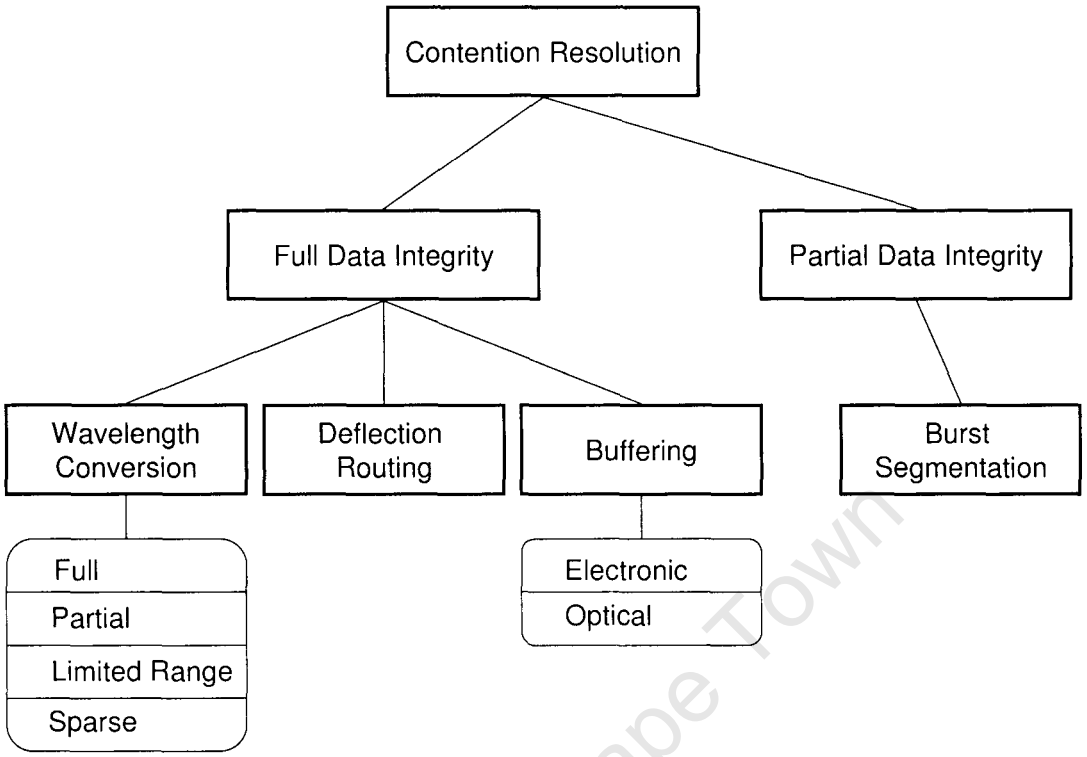


Figure 2.14 OBS Contention Resolution Techniques.

Wavelength conversion, deflection routing and buffering maintain full data integrity while burst segmentation maintains partial data integrity. Burst segmentation is used as a last resort to keep the maximum amount of data instead of dropping a burst. Wavelength conversion can be used to different extents. Because the cost of tunable wavelength converters is high, researchers have to find ways to provide efficiency while maintaining the cost-effectiveness of an OBS network. Therefore, the use of wavelength conversion can be partial, limited-range and sparse. Buffering can be electronic or optical. Electronic buffering is conventionally present at the edge of the network whereas optical buffering is used in the core network. Deflection is the redirection of bursts to an alternative path in order to reach the same destination. The alternative route is a secondary option and may increase the end-to-end delay of the burst/packet.

2.5.1 Wavelength Conversion

Tunable wavelength converters (TWCs) are devices that allow wavelength conversion at routers in an OBS network. Wavelength conversion is the process that switches a burst from its input wavelength channel to any output wavelength channel on the same link. The increase in the possible number of wavelengths per link (160-320 wavelengths per fiber in the near future), makes wavelength a primary option when resolving contention. We classify wavelength conversion as follows:

- Full conversion: Any incoming burst can be shifted from any input channel to any output channel.
- Limited conversion: The shifting of wavelengths for each input channel is restricted to a limited number of output channels. Thus, the cost of the switches is reduced at the expense of higher blocking.
- Fixed conversion: This is a restricted type of limited conversion where each input channel can only be connected to one or more pre-determined output channels.
- Sparse conversion: the network can have a collection of nodes that have full, limited, fixed and no wavelength conversion. The use of algorithms can minimize the number of wavelength converters in an OBS network.

The most efficient scheduling algorithms are based on the assumption of a full wavelength conversion capability of the network. Although this option is not cost-effective, it provides the network with an improved flexibility and several performance benefits that are needed to reduce burst loss in the network.

2.5.2 Optical Buffering (Fiber Delay Lines)

Due to the lack of optical RAM (random access memory), buffers called Fiber Delay Lines (FDLs) were implemented to delay the arrival of bursts at intermediate nodes. FDLs can be advantageous because they can offer lower latency than the delay suffered in dropping, redirecting, and retransmitting bursts when contention

occurs. On the other hand, delaying a single burst for a 1ms requires over 200km of fiber. FDLs can therefore only provide limited storage capacity. Furthermore, FDLs are needed for every wavelength in the fiber. They create more voids on the wavelength channels and therefore increase the complexity of scheduling algorithms. This added complexity causes scheduling algorithms to run slower and therefore reduces the performance of the network in terms of delay constraints and burst loss rate. Because the FDL technology is not cost-effective and offers limited benefits, it is acceptable in prototype switches but is not viable in the industry. Electronic buffering, often used at edge nodes for legacy networks, can be combined with optical buffering. However, this combination would be at the expense of higher network costs and higher complexity in the network. In this thesis, we assume an OBS network with no FDL buffers.

2.5.3 Deflection Routing

To resolve contention with deflection, the core router switches a burst to an output port different from its designated one. Although deflection routing has been studied for electronic and optical packet switched networks [23-25], there is limited work that applies deflection to OBS networks. Deflection can be applied in a wavelength, time, and/or space domain.

- Deflection in the time domain refers to the use of FDLs (Section 2.5.2) to delay the contending burst.
- In the wavelength domain, wavelength conversion is used to switch a contending burst to another wavelength in the same fiber and thus, improves the flexibility of the network [6].
- Space domain deflection allows a contending burst to be switched to a different output port and then follows an alternate route to destination. Following an alternate route may cause the contenting burst to arrive late at destination especially if the burst is deflected several times.

While using deflection routing in an OBS network can cause bursts to arrive late at their destination, it improves the flexibility of the network. It can improve

network performance if used effectively (i.e. limited number deflections). Thus, we assume the use of deflection routing in our OBS network.

When deflection is not possible because no output port, wavelength, or FDL is available, the loss of data is unavoidable. The contending data burst is therefore dropped. In the case where priorities or traffic profiles are assigned to bursts, it is possible for a contending burst to preempt an original burst.

2.5.4 Burst Segmentation

The concept of burst segmentation [26, 27] aims at reducing *packet* loss when data loss is unavoidable. To implement burst segmentation, [26, 27] use Time Division Multiplexing (TDM) to divide bursts into segments of fixed size. Each segment consists of one or several packets and defines partitioning points when segmentation is necessary (Figure 2.15). When contention occurs, instead of dropping a whole burst, the overlapping segments of a given burst with another are segmented. These segments are then dropped, deflected, or preempted.

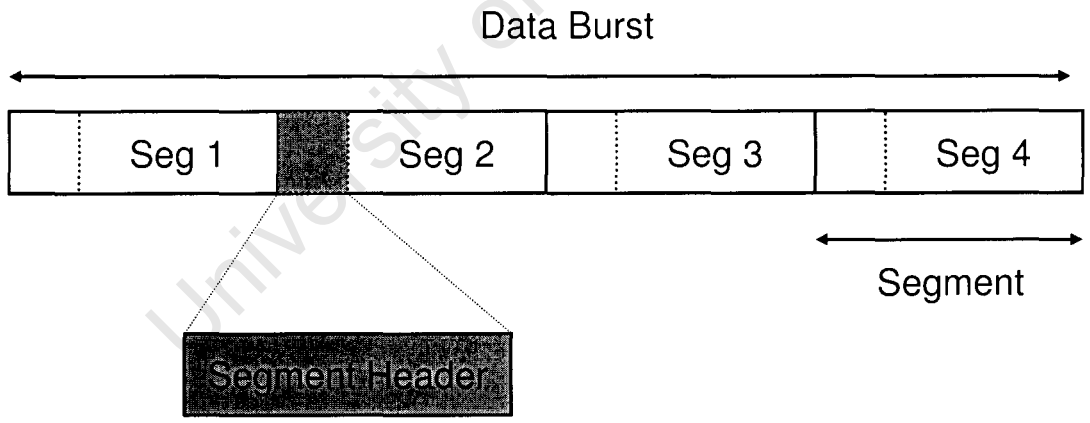


Figure 2.15 Burst format for Burst Segmentation.

There are two approaches to segmenting bursts when contention occurs. The first approach is tail dropping and consists in segmenting the tail of the original burst. Head dropping is the second approach and consists in segmenting the head of the contending burst. Under the assumption that bursts are retransmitted, the tail dropping approach has a better chance of delivering packets in the correct order than

the head dropping approach. Thus, we only consider the tail dropping approach. Figure 2.16 reveals the tail dropping approach.

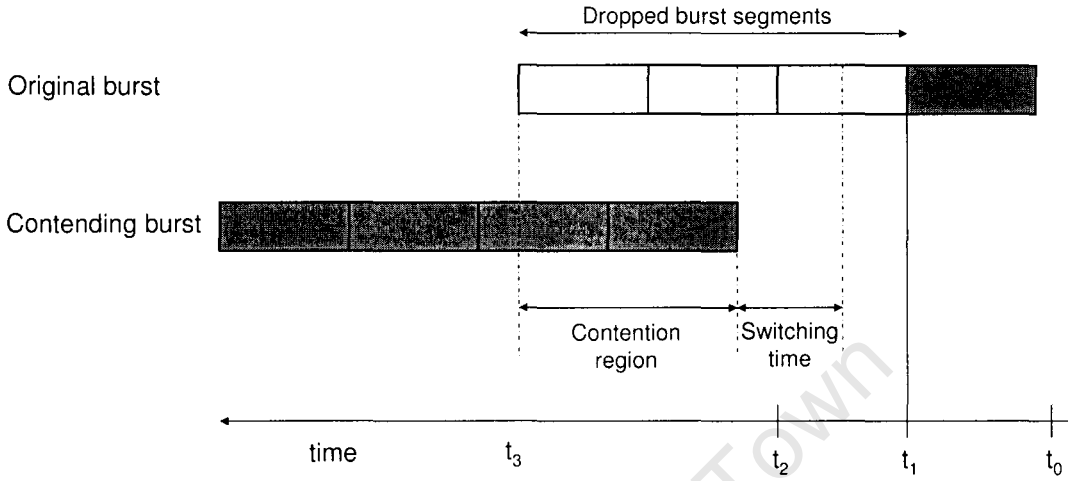


Figure 2.16 Tail Dropping Approach in Burst Segmentation.

An important parameter to consider in burst segmentation is *switching* time. The switching time is the time needed to reconfigure the node for burst segmentation. As shown in Figure 2.16, if the switching time is negligible (fast), the original burst would be segmented at t_2. On the other hand, if the switching time is not negligible, more packets are lost because the original burst is segmented at t_1. The switching time is therefore a direct measure of packet loss in burst segmentation [26].

Burst segmentation can be combined with other contention resolution schemes to provide better QoS in the network [26, 27]. [27] proposed to implement segmentation with deflection. Instead of dropping bursts when contention occurs, the burst can be deflected or its segmented tail can be deflected. These options increase the chance that a packet will reach its destination and thereby improve network performance in terms of throughput. However, this improved performance may not apply in the case of delay sensitive applications. The following combinations are used to resolve contention with burst segmentation [27]:

1. *Segment First and Deflect Policy (SFDP):* The contending burst wins

the contention and the original burst is segmented. The segments of the original burst may be deflected if an output port is available otherwise it is dropped

2. *Deflect First and Drop Policy (DFDP):* The original burst wins the contention. The contending burst is deflected if an output port is available otherwise it is dropped.

3. *Deflect First Segment and Drop Policy (DFSDP):* The node attempts to deflect the contending burst to a free output port. If no port is available, original burst is segmented and its tail is dropped while the contending burst is transmitted

Although the combination of burst segmentation with deflection routing can improve the network performance in terms of throughput, deflecting the tail segment of a burst has several disadvantages:

- The tail segments of bursts are deflected to alternate ports as newly created bursts. Creating new bursts require new burst headers and increase the processing complexity of the node. This leads to high control overhead with respect to switching times
- Tail segments may be deflected numerous times in the network. Hence, bandwidth resources are wasted and contention is increased within the network.

Using Time Division Multiplexing at the edge nodes to implement burst segmentation complicates network control. Although it may improve network throughput, it creates a high control overhead at every core node in the network. Furthermore, the point of OBS is to reduce packet-processing time at the core nodes with burst assembly at the edge nodes. Burst segmentation is therefore in contrast with the assumption of transparency and simplicity of OBS. Segmentation-based contention resolution will not be considered for the rest of thesis.

Table 2-3 presents a summary of the main advantages and disadvantages of each contention resolution method. These contention resolution methods seem to complement each other's strengths and weaknesses. Hence, it may be preferable to combine them to achieve better overall network performance.

Table 2-3 Comparison of OBS Contention Resolution Techniques.

Contention Resolution Scheme	Advantages	Disadvantages
Wavelength Conversion	Low burst loss	Immature and expensive
Fiber Delay Lines	Mature technology; conceptually simple	Bulky FDLs; more voids; Extra Delay; Limited Storage
Deflection Routing	No extra hardware required	Late arrivals
Burst Segmentation	Improved throughput	Complicated control

Recent times have shown that network traffic is diverse. Applications have unique requirements in terms of bandwidth, delay, and loss. A substantial issue in next-generation networks such as OBS is the capacity to support the requirements of these applications. The network should therefore be able to provide some level of guarantees with respect to these performance parameters. In the next section, we define Quality of Service (QoS) and Quality of Experience (QoE). In addition, we describe and compare the current ways used to support QoS in OBS networks.

2.6 Quality of Service in OBS Networks

Quality of Service (QoS) is a term used to describe the overall experience that a user or an application will receive over a network. To this day, the explosive growth of the internet is fuelling a rapid increase in the demand for more internet bandwidth. Although optical networks can provide high bandwidth capacities, they also have to provide high end-to-end QoS. Because electronic routers are the

bottleneck in optical networks, their performance has a significant effect on end-to-end QoS in terms of packet loss, delay and throughput. Furthermore, network operators need to ensure that users receive good Quality of Experience (QoE) when using applications.

QoE is the perception of the users on how well a system or an application performs in relation to their expectations. QoE also includes how well the user can intuitively use the application in a timely and efficient manner with no worries of the underlying network elements. Although QoE and QoS are related, they are not the same. It is possible to have excellent QoS but have poor QoE, as in the case of the flawless transmission of corrupted packets. The main difference between QoS and QoE is that QoS is measured objectively whereas QoE is a subjective measurement which needs to be translated in quantitative terms. Although network operators rely on QoS metrics to determine the level of quality to use for different services, it is critical to incorporate the use of QoE as part of our engineering methods. Hence, we can ensure a customer oriented perspective that can help us move beyond exclusive network performance QoS metrics [28]. We must therefore select efficient QoS mechanisms to satisfy the end-user QoE for any given application. In this way, we can create a bandwidth efficient and QoS-enabled OBS network that supports various types of applications and provides good QoE to all users.

Figure 2.17 reveals the classifications of QoS mechanisms in an OBS network. QoS can be provided from either the control plane or the data plane of an OBS network. In the control plane, QoS is offered through the means of routing and signaling. While in the data plane, the edge and core router provide the service differentiation.

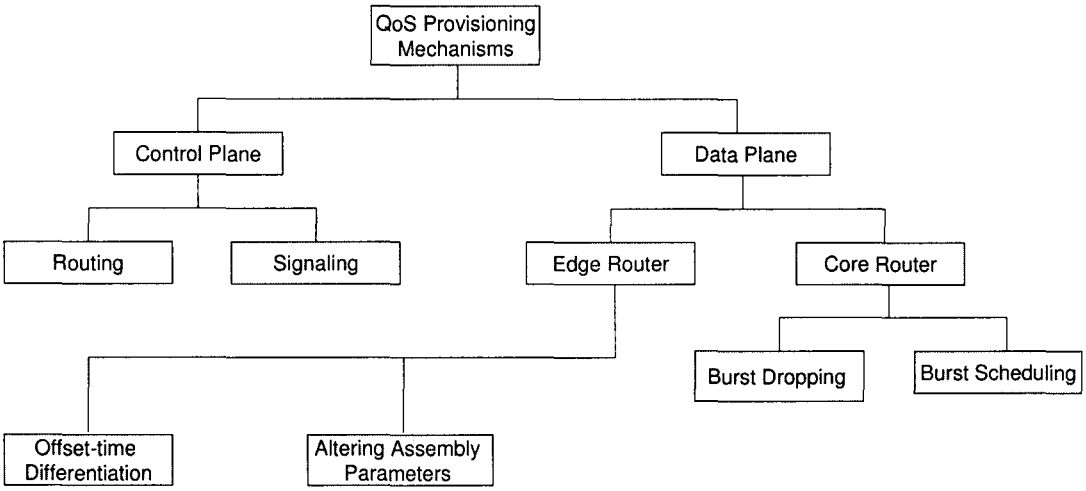


Figure 2.17 Classification of QoS Mechanisms.

OBS uses one-way reservation protocols (JET, JIT), which performs according to a statistical multiplexing paradigm. This type of signaling protocol favors the need for lower end-to-end transfer latency at the expense of high loss in the network. Thus, additional QoS support is needed to meet the demands of loss-sensitive traffic. In this way, high profile traffic is differentiated from low profile traffic.

In an effort to provide service differentiation in IP over WDM, several models were proposed, notably IntServ [29] and DiffServ [30]. The IntServ model, based on a per-flow classification of traffic, proved to be too rigid and non-scalable for large networks. On the other hand, the DiffServ model, which was based on a per-class classification of traffic, offered the scalability, flexibility that was lacking in IntServ. DiffServ is well suited for OBS because complex operations are pushed to the edge while the core is kept as simple as possible.

We distinguish between two DiffServ models of QoS within an OBS network: *relative service differentiation*, also referred to as *relative* QoS and *absolute service differentiation*, also referred to as *absolute* QoS. In the relative QoS model, the performance of the network cannot be guaranteed in quantitative terms. Instead, the QoS parameters of a certain class are given relative to another class. For instance, the burst loss of high priority bursts is guaranteed to be lower than that of low

priority bursts. However, the loss of high priority bursts still depends on the traffic load of low priority bursts. Thus, there is no guarantee on the upper bound of the loss probability of high priority traffic [9]. On the other hand, absolute QoS offers a worst-case guarantee to different traffic types and offers an upper bound on QoS metrics. The provision of hard guarantees supports applications with stringent delay and loss requirements.

2.6.1 Relative QoS

In the OBS network, relative QoS is usually offered in the data plane, at the edge nodes. Relative QoS can also be applied at the core nodes, at the expense of higher complexity and higher processing delays. At the edge nodes, static QoS is offered with burst assembly. Here, packets are differentiated based on their class and destination. Specific attributes such as labels or priorities can be assigned to them for further discrimination at the core of the network. QoS mechanisms at the edge nodes include *Offset-Time Differentiation* [31], and *Variable Timer-based Assembly* [32] or *Burst Length Differentiation* [33]. In the core network, other ways to provide relative QoS are preemptive dropping [34] and threshold-based dropping [35]. These core mechanisms can be extended to provide absolute QoS.

Offset-Time Differentiation based QoS [31]

Offset-Time differentiation was the first approach to providing QoS in an OBS network. The basic idea of the offset-time based QoS mechanism is to assign different offset times to different burst classes. High priority bursts are assigned extra offset times and thus gives them a higher chance of being scheduled without conflict at the core nodes. In this way, the loss of high priority bursts is minimized. The extra offset assignment further eliminates the need for differentiation mechanisms within the core network. Figure 2.21 illustrates a case where the Offset-Time based differentiation scheme is advantageous.

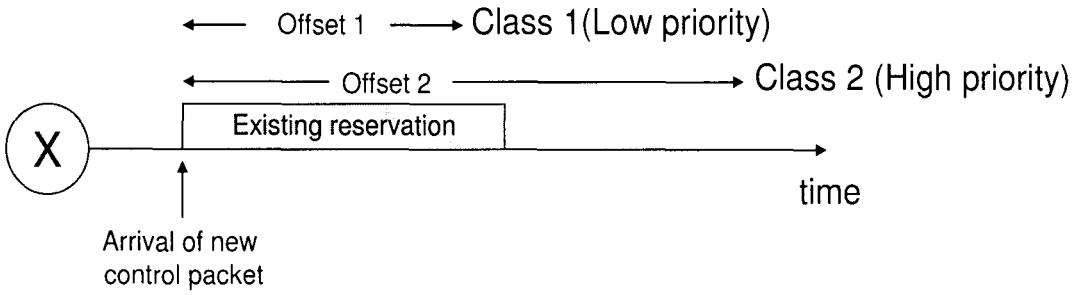


Figure 2.18 Offset-Time Differentiation based Assignment.

When the incoming burst has a low priority, there is contention with the existing reservation (Figure 2.18). On the other hand, if the incoming burst has a high priority, there is no contention. The high priority burst is scheduled because it has a large offset time. Hence, it can be scheduled after the existing reservation (Figure 2.18). The drawback of this scheme is that high priority bursts will experience long delays due to the extra offset time. Thus, the offset-time based QoS scheme may satisfy loss requirements but cannot meet delay requirements of high priority bursts. Furthermore, [36, 37] showed that this scheme can lead to unfairness as large low priority bursts will experience higher loss than small low priority bursts.

Variable-timer based Assembly [32] or Burst Length Differentiation [33]

Variable-timer based Assembly or Burst length differentiation is based on assembling packets with varying assembly periods. The assignment of the assembly time is based on the delay requirements of the incoming packets at the edge nodes. Because of the assembly delay incurred at the edge of the network, it is important to shorten the period of assembly of high priority packets. A short assembly period increases the probability of meeting the stringent delay requirements of high priority packets. This short assembly time is at the expense of an increased number of control packets in the network. On the other hand, the short assembly times mean that high priority bursts will be smaller and will have a higher chance of being scheduled in the core. Furthermore, packets with low priority are assembled for longer periods. Hence, low priority packets will form larger bursts than packets with

high priority. The effect of the longer assembly time is a reduced number of control packets in the network. A balance is therefore created in limiting the possibility of high control overheads in the core due to an excessively high number of control packets. The benefit of variable assembly periods is an improvement in the blocking probability of high priority bursts as well as meeting their stringent delay requirements. The drawback of variable assembly periods could be the requirements for higher switching times due to shorter bursts.

Preemptive dropping [34]

This scheme is present in the data plane, at the core nodes. Preemptive dropping consists in overwriting the resources used to reserve a low priority burst with a high priority burst, in case of contention. The preempted low priority burst is then discarded. The pre-emption can be either full or partial (Figure 2.19). The partial pre-emption consists only in pre-empting the section of the low priority burst that is contending with the high priority burst. Partial pre-emption increases complexity in the burst assembly process at the edge as well as in the core node. This pre-emption mechanism leads to fine class isolation but leads to a waste of resources for the preempted burst over consecutive nodes. Furthermore, a signalling protocol may be needed to release the resources of the preempted burst and thus increases core complexity.

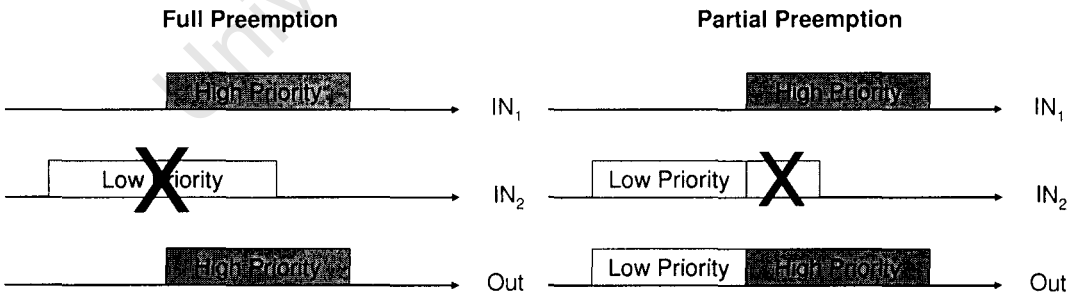


Figure 2.19 Preemptive Dropping.

Threshold-based dropping [35]

Threshold-based dropping is a scheme that limits the availability of wavelengths or a buffer (in the case of FDLs) to specific traffic classes. For example, if a link has four wavelengths channels available, low priority bursts can only be scheduled on two channels. High priority bursts on the other hand, can be scheduled on any of the four channels (Figure 2.20). In this way, high priority bursts have more resources than low priority bursts. The benefit of the threshold-based scheme is its easy implementation while providing differentiation at the core. Nevertheless, its efficiency strongly depends on its threshold adaptability to actual traffic.

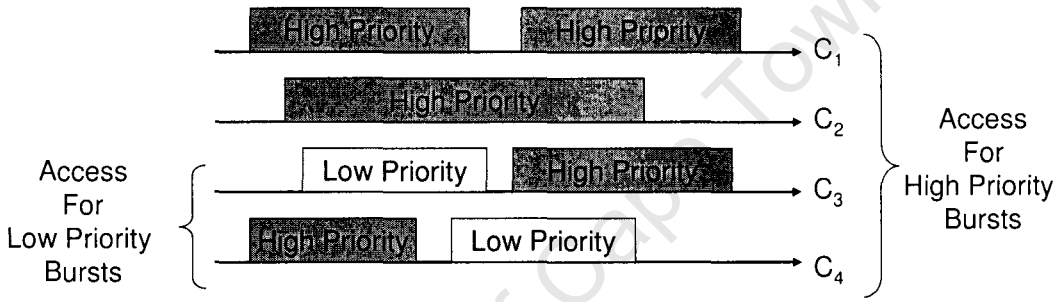


Figure 2.20 Threshold-based Dropping.

In summary, the strength of relative differentiation is its simplicity and flexibility. On the other hand, the weakness of relative differentiation is its lack of QoS guarantees. We now describe absolute QoS mechanisms that offer guaranteed services in the network.

2.6.2 Absolute QoS

The intuitive way to provide absolute QoS in OBS would be to reserve specific wavelengths for high priority traffic in an optical circuit-switching manner. This approach can be seen as a hybrid signalling protocol that consists in combining the two-way and one-way reservation modes. End-to-end wavelength paths could provide guarantees such as no losses and negligible delays (two-way reservation). On the other hand, best effort traffic would use the unreserved resources (one-way reservation). This approach can provide absolute QoS at the expense of inefficient

bandwidth utilization. An objective of OBS is to increase bandwidth utilization to service a maximum number of users. Hence, other efficient methods are needed to provide absolute QoS guarantees in the network.

The essence of providing absolute QoS lies in differentiating between traffic in a probabilistic manner. Hence, the majority of relative QoS mechanisms can be extended to provide absolute QoS. The provision of absolute QoS comes at the expense of online measurements of loss probability. Hence, each OBS core node needs to monitor traffic statistics such as the burst arrivals and burst drops for each guaranteed class. Recent Absolute QoS mechanisms are based on Probabilistic Preemption [38], Intentional Burst Dropping and Wavelength Grouping [35].

Probabilistic Preemptive QoS [38]

Probabilistic preemptive QoS in [38] is designed to provide service differentiation in terms of burst blocking probability. High priority bursts are assigned a preemptive probability that allows them to preempt low priority bursts in a probabilistic manner. Furthermore, the ratio of burst blocking probability can be modified between classes without affecting the overall burst blocking probability [38].

Intentional Burst Dropping

Intentional burst dropping is a technique used to maintain the loss guarantees of each traffic class in the network. Authors in [35, 36] have proposed various ways to implement intentional burst dropping in an OBS network using Random Early Detection (RED). RED maintains the performance requirements of high priority bursts at the expense of lower priority bursts. Hence, RED intentionally drops low priority bursts to achieve the loss guarantees of high priority bursts. This approach can therefore guarantee absolute QoS. In contrast to offering absolute QoS, intentional burst dropping can lead to low link utilization. Furthermore, the implementation of this approach can be complex, as it requires constant online measurements of loss at each core node. Hence, network performance at heavy loads

may drop considerably due to longer processing times. The longer processing times will also affect the switching times of bursts.

Wavelength grouping [35]

Wavelength grouping is a variation of the threshold-based dropping scheme used for relative QoS. Wavelength grouping consists in assigning a set of wavelengths to specific traffic classes. In other words, each class has its own set of pre-assigned wavelengths for scheduling. In the case of Static Wavelength Grouping (SWG), the wavelength assignment is respected even if a burst cannot be scheduled on its pre-assigned wavelengths. Dynamic Wavelength Grouping (DWG), on the other hand, offers greater flexibility. If the router cannot schedule a specific burst on its pre-assigned wavelengths, it schedules it on any other available wavelength. This flexibility comes with a higher complexity of implementation than for SWG.

In brief, although absolute QoS mechanisms can provide guarantees, they require complex implementations in the core network. These complex techniques may turn out to be too rigid to accommodate for the ever-changing types of applications and user QoE requirements. Furthermore, relative QoS mechanisms provide good performance while maintaining simplicity and transparency in the core, which is the essence of next generation networks - pushing the intelligence to the edge of the network. A good way to provide overall QoS may be to combine the use of both Absolute and Relative QoS mechanisms. This combination will help limit the complexity of the core network while providing QoS guarantees and good overall QoE for users.

3 Enhancing QoS in OBS Networks

In this chapter, we highlight specific contention resolution issues that affect QoS in OBS networks, and propose a scheme to enhance network performance. The remainder of this chapter is organised as follows. Section 3.1 evaluates specific QoS issues in OBS and describes the main objective of this thesis. Section 3.2 proposes a scheme to enhance QoS in OBS and illustrates its application to burst assembly and contention resolution. Section 3.3 describes the important metrics used to evaluate the performance of an OBS network, and Section 3.4 lists the predicted effects of the proposed scheme on network performance.

3.1 QoS issues in OBS

A major concern in OBS is the efficient and cost-effective resolution of contention. The loss of bursts due to contention needs to be minimized in order to increase bandwidth utilization and thus improve network efficiency.

Currently, the best method to resolve contention is wavelength conversion. Wavelength conversion makes use of multiple wavelengths to reduce contention. A burst may be switched from any input wavelength to any available wavelength on the outgoing link using Tunable Wavelength Converters (TWCs). Unfortunately, the high cost of TWCs means that operators have to limit its use in the network to remain cost-effective. Hence, the combination of wavelength conversion with other methods of contention resolution may be necessary to reduce burst loss in the core network.

In electronic packet-switching networks, contention is usually resolved through buffering. However, buffering capabilities are limited in the optical domain. The use of fiber delay lines is not practical and offers limited storage capacities.

With burst segmentation, the loss of entire bursts is prevented when data loss is unavoidable. Each burst is made up of segments and only the overlapping segments in contention are dropped. Hence, packet loss is reduced and throughput is increased. However, burst segmentation creates a high control overhead in the core network. This high control overhead goes against the assumption of transparency in the core network. Furthermore, segmentation increases the number of short bursts in the network. Switching short bursts requires faster switching technology, which is one of the limitations of implementing optical packet switching.

In deflection routing, a contending burst is switched on a different output other than the intended output port. The main advantage of deflection routing is that it requires no extra hardware for implementation, and is therefore cost-effective. On the other hand, it has several shortcomings. Deflected bursts suffer larger propagation delays in the network. This extra delay affects the end-to-end transmission delay of bursts. Hence, bursts may reach their destination late, and packets may be delivered out-of-sequence. It has been shown that deflection routing increases throughput at low loads [5]. However, at high loads deflection routing has a negative effect because a higher number of deflections increase the probability of contentions [5]. Hence, the probability of burst loss increases. Furthermore, there is an issue of offset time assignment when deflection routing is used in the network. The offset time assigned at the edge of the network needs to be sufficient to prevent the payload (burst) from catching up with the control packet. Thus, if a large number of deflections occur and the offset time is not large enough, the payload can catch up with its control packet and unnecessary burst drops may occur.

Deflection routing is not a preferred method in electronic switched networks. However, it may be necessary to implement deflection routing in OBS networks due to their limited buffering capacity.

Eliminating the major shortcomings of deflection routing can make it a suitable candidate to support wavelength conversion as a method of contention resolution in OBS networks. Hence, the main objective of this thesis is to reduce the

negative impact of burst deflections on late packet arrivals and to improve the goodput of deflection routing. For this purpose, we propose an Emission and Discard Priority (EDP) scheme in the next section.

3.2 An Emission and Discard Priority Scheme for QoS Support in OBS Networks

To accommodate the QoS requirements of internet traffic and to overcome the limitations of deflection routing, we introduce the concept of emission and discard priorities (EDP) in OBS. A good way to provide QoS is to have a service differentiation scheme that discriminates effectively between traffic types in order to meet their QoS requirements. The EDP scheme provides service differentiation at both the edge and the core by focusing on the performance requirements of current applications. As a result, the QoS constraints of traffic can be met, and more importantly, the Quality of Experience (QoE) of users can be improved. Figure 3.1 illustrates the major applications present in internet traffic and their sensitivity to different QoS metrics.

Performance dimensions				
Application type	Bandwidth	Sensitivity to		
		Delay	Jitter	Loss
VoIP	Low	**High**	**High**	Med
Video Conferencing	**High**	**High**	**High**	Med
Streaming Video	**High**	Med	Med	Med
Streaming Audio	Low	Med	Med	Med
Email	Low	Low	Low	**High**
File Transfer	Med	Low	Low	**High**

Figure 3.1 Performance Constraints of Different Types of Applications.

The emission priority determines the urgency of delivery of incoming traffic. This priority is based on the delay tolerance of each application type. For example,

Figure 3.1 shows that VoIP and video conferencing applications, which are highly sensitive to delay, will have a high emission priority. On the other hand, application such as Email and File transfer, which are highly tolerant to delay, will have a low emission priority. Hence, traffic with a higher emission priority has precedence over traffic with a lower emission priority.

The discard priority determines the order in which bursts are discarded. Bursts are discarded either when contending for resources, or if their traffic class is out of profile when using absolute QoS mechanisms. The discard priority gives further differentiation in the core network. We differentiate between bursts of equal emission priorities with the discard priority in the case of contention. Bursts with higher discard priorities are more eligible to be dropped than bursts with lower discard priorities. The advantage of discard priorities is that they create virtual queues at the edge nodes. Hence, a low number of hardware queues can still provide a high number of QoS levels.

The discard priority of a packet is based on its delay and loss requirements, as well as its transmission protocol. For example, interactive applications are UDP-based (Universal Datagram Protocol) and hence cannot retransmit lost or dropped packets. Furthermore, packet retransmission would be useless because interactive applications are real-time based. On the other hand, responsive applications are both UDP-based and TCP-based (Transport Control Protocol) and further have buffers to improve QoE. Hence, packet retransmission is possible and useful because the requirements of responsive applications are near real-time. The retransmission of timely applications is also possible because they are TCP-based. We can therefore say that the transmission protocol is an important factor in determining the discard priority of packets.

3.2.1 Burst Assembly with the Emission and Discard Priority (EDP) Scheme

When packets arrive at the edge of the network, they are aggregated in separate queues based on their destination and delay requirements. The delay

requirements of the incoming traffic determine the emission priority of the assembled packets. Hence, the emission priority determines the importance of the burst in the network. The emission priority therefore provides static QoS classification at the edge node. A high emission priority means a high burst class. Figure 3.2 shows the format of a burst header packet (BHP) when assembling with the EDP scheme. We can see that there is an emission priority field and a discard priority field in the burst header packet. These fields are read when the burst header packet reaches a core node. In the case of contention with another incoming burst, the core node uses the values in the EP, DP, and BL fields to resolve contention (Figure 3.2). The core node uses the rest of the fields to determine whether the incoming burst can be reserved on an outgoing link.

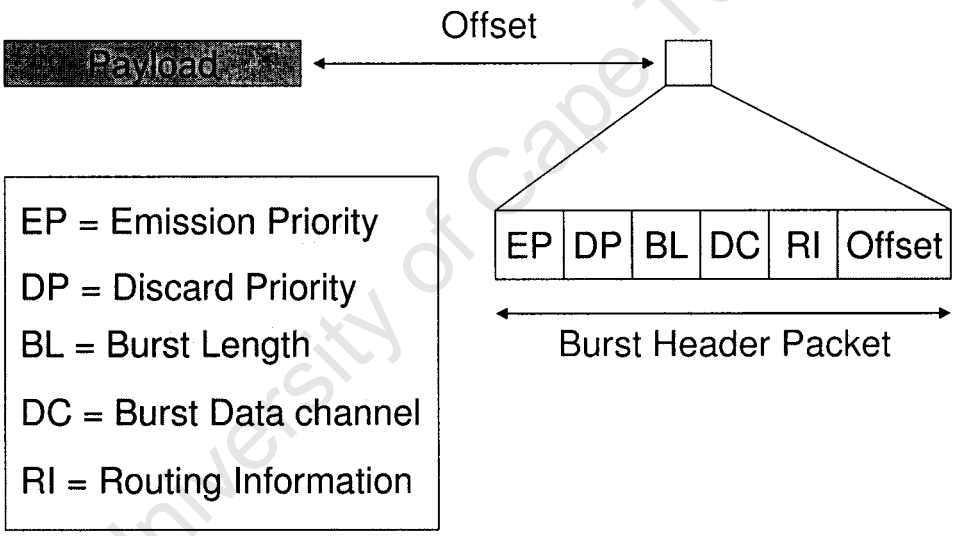


Figure 3.2 Format of Burst Header Packet with the EDP Scheme.

In our OBS network, we assume that the assembly period of packets at the edge nodes is variable [32]. Hence, the emission priority also dictates the assembly period of each assembly queue. A high emission priority implies a short assembly period whereas a low emission priority implies a longer assembly period. However, short assembly periods imply short bursts and short bursts imply higher switching time requirements. Hence, the padding of short bursts may be needed to ensure that their length/size is within the switching capabilities of the core nodes in the network.

3.2.2 Resolving Contention with the Emission and Discard Priority Scheme

As mentioned in Section 3.2.1, each burst has an emission priority and a discard priority. Figure 3.3 illustrates the QoS classification of internet traffic applications. The emission priority corresponds to the traffic category and the number of hardware queues at every edge node. Discard priorities allow applications that are within the same traffic category to be subdivided. In this case, applications in the same traffic category are subdivided with two discard priorities. For example, in the interactive class, VoIP applications have a lower discard priority than interactive gaming and video conferencing applications. Because telephone companies have set the real-time standards for telephony for over 100 years on their circuit switched networks, VoIP has higher QoE requirements.

Traffic Category	Applications	Emission Priority	Discard Priority
Network Control	Critical alarms	3	0
	Critical OAM, Routing, Billing		1
Interactive	VoIP	2	0
	Interactive Gaming Video Conferencing		1
Responsive	Streaming Video, Audio	1	2
	Client/Server Transactions		3
Timely	Email, non-critical OAM	0	3
	Best Effort		4

Figure 3.3 QoS Service Differentiation with EDP Scheme.

At any edge node in the network, each queue can only assemble a burst with one emission priority and one discard priority. Therefore, even though packets may have different discard priorities, the burst that they form must have one discard

priority. This issue can be resolved in several ways. For example, a burst can either be given the lowest discard priority present in its group of packets, or be given the discard priority that occurs the most in its group of packets.

Table 3-1 illustrates the possible cases of burst contentions when using the EDP scheme as a guide to resolve contentions. We attempt wavelength conversion for both contending bursts before attempting deflection routing. Deflection routing is attempted for the losing burst. In case the emission and the discard priorities of contending bursts are equal, we use the burst length to differentiate. The shorter burst is deflected because it offers potentially lower throughput; thus, higher priority is given to the longer burst.

Table 3-1 Contention Cases using the EDP scheme

Contention	Emission Priority	Discard Priority	Burst Length	Deflection
Case 1	Burst A > Burst B	X	X	Burst B
Case 2	Burst A < Burst B	X	X	Burst A
Case 3	Equal	Burst A > Burst B	X	Burst A
Case 4	Equal	Burst A < Burst B	X	Burst B
Case 5	Equal	Equal	Burst A > Burst B	Burst B
Case 6	Equal	Equal	Burst A < Burst B	Burst A

3.2.3 Optimizing Deflection Routing with the Emission and Discard Priority Scheme

Although several authors [5, 39, 40] have studied and analysed deflection routing, they were mostly concerned with the burst blocking effects of deflection routing. In this thesis, we focus on the more important disadvantage of deflection routing, which is the added delay that bursts suffer when deflected. Although

deflection routing will increase throughput in the network, it will not necessarily improve the goodput of the OBS network. Goodput can be defined as the useful throughput that reaches destination. Packets with stringent delay tolerances that arrive late at their destination are useless. Therefore, these packets should not be considered as throughput because they do not contribute to providing QoS or QoE to the end user. It is preferable to drop the bursts that contain these packets inside the network to release resources for other incoming bursts.

In this section, we describe a way of improving the goodput of the network using the emission and discard priority schemes. The EDP scheme helps to limit the deflection of bursts that have a high risk of reaching their destination beyond the limits of their delay requirements. Figure 3.4 shows the deflection routing algorithm that we use to improve goodput in the OBS network. The discard priority determines the number of possible deflections that a burst can suffer. When a burst is deflected, we decrement its discard priority. Hence, if discard priority of a burst equals zero, the burst cannot be deflected. Furthermore, a burst with a lower discard priority has a higher chance of winning in the case of contention. Thus, when a burst is deflected and its discard priority is consequently decremented, its importance in the network increases. As a result, we can potentially enhance the chances of a burst reaching its destination within its delay limits.

As shown in Figure 3.3, timely bursts have a high discard priority whereas network control and interactive bursts have a low discard priority. Hence, delay tolerant applications, which are usually loss intolerant, can spend more time in the network to ensure delivery. On the other hand, delay intolerant bursts cannot afford many deflections because of time constraints. The overall effect is that the number of unnecessary deflections is reduced and goodput of the network is improved. This effect may not be obvious at low loads due to the low probability of contention. However, this effect can be noticeable at high loads where unnecessary deflections cause more contentions in the OBS network.

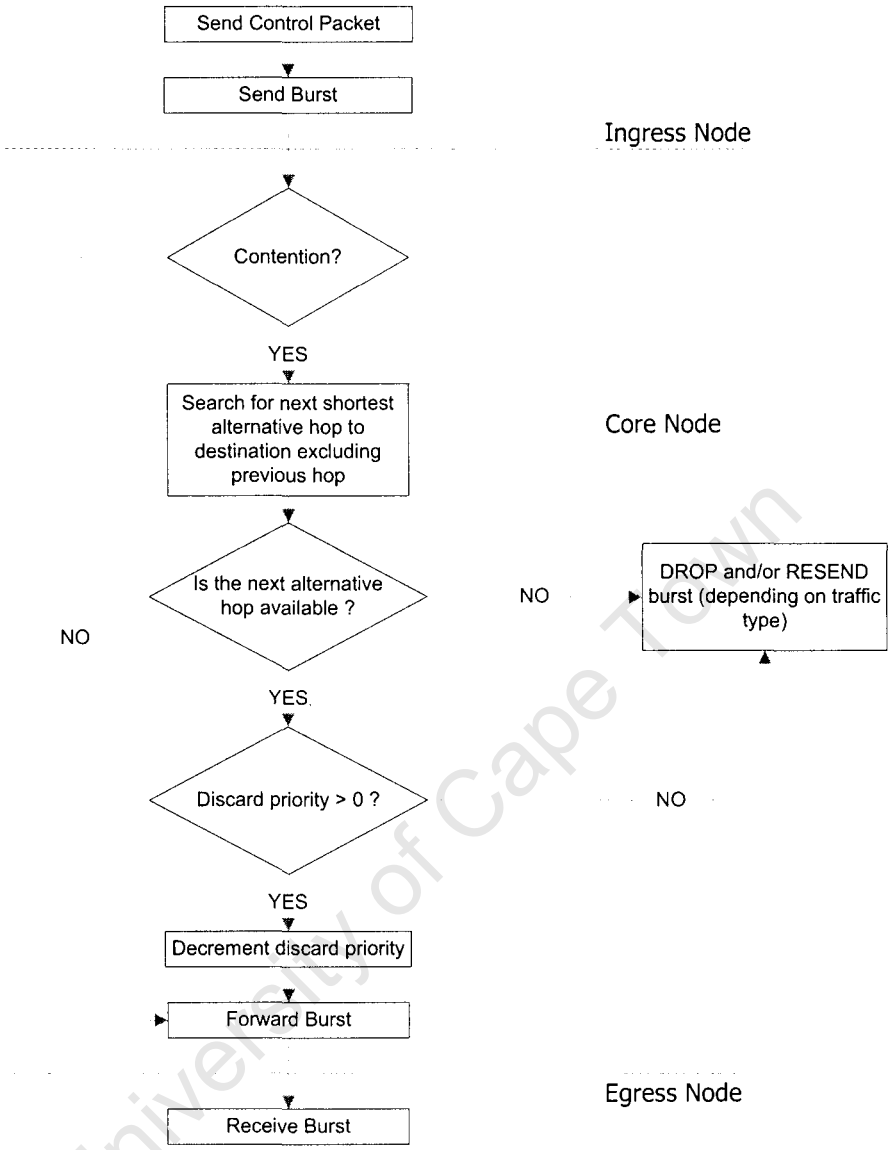


Figure 3.4 Flow Diagram of Deflection Routing combined with the use of the EDP Scheme.

We implemented deflection routing under the assumption that deflection is only attempted on the next shortest alternative hop to destination. If the burst cannot be scheduled on the next shortest alternative hop, it is dropped. In this way, we ensure that deflection routing does not cause a high contention probability in the network. Furthermore, we can say that deflecting bursts becomes a bonus because it is used only when there is an available next hop. To evaluate the performance of

deflection routing with the EDP scheme, we need to analyse the parameters used to measure performance in OBS networks.

3.3 Performance Metrics in OBS networks

To evaluate QoS, we define some key QoS metrics that reflect the performance of our OBS network at different traffic loads. These QoS metrics are as follows.

Throughput – is the amount of data delivered per time unit over a physical or logical link. Throughput is measured in bits per second (bps), and is always less than or equal to the link/channel capacity. The maximum throughput of a link/channel is just its capacity.

Total End-to-end Delay (T_{delay}) - is the time elapsed from the arrival of a packet at the edge router until the delivery of the packet to its destination router. The total delay (T_{delay}) is the sum of the burst assembly time ($T_{assembly}$), the offset time (T_{offset}), the transmission time (T_{tx}), the propagation time (T_{prop}), the switching time (T_{switch}) and the burst disassembly time ($T_{disassembly}$). Hence,

$$T_{delay} = T_{assembly} + T_{offset} + T_{tx} + T_{prop} + T_{disassembly} + T_{switch}$$

$T_{disassembly}$ and T_{switch} are negligible factors in the overall end-to-end delay. Deflection routing affects the burst propagation time. Therefore, reducing T_{prop} is crucial to improving the performance of deflection routing as a contention resolution technique.

Burst blocking probability (P_{loss}) – is the probability of burst loss in the network. We define it as

$$P_{loss} = 1 - \frac{\text{no. of bursts received}}{\text{no. of bursts sent}}$$

Goodput – is the amount of useful data received per unit time at the destination router. In our OBS network, we define goodput as the ratio of packets

that reach their destination within their delay requirements. Goodput is always less than or equal to the throughput. To improve the overall QoS and QoE of users, improving goodput is just as important as improving throughput.

3.4 Predicted Effect on Network Performance

The predicted effects on the network performance when using deflection routing with the EDP scheme are as follows:

- Simple and effective service class differentiation at the edge and the core of the network.
- Higher throughput than the drop policy where deflection routing is not implemented as contention resolution scheme.
- Higher throughput and higher goodput than deflection routing without the EDP scheme.
- Lower end-to-end delay due to the limited number of unnecessary deflections, which are based on traffic requirements.

4 Experiment Model and Network Setup

In order to evaluate the performance of the emission and discard priority (EDP) scheme, we extended an OBS network simulation tool. In this chapter, we describe the simulation environment, present our simulation objectives and explain the methodology used to evaluate the EDP scheme in an OBS network.

4.1 Simulation Environment

For our simulations, the simulation environment shown in Table 4-1 applies.

Table 4-1 Simulation Environment for OBS simulations

Computer Processor	Intel Core Duo 1.66GHz
RAM	1GB
Operating System	Fedora Core 4 Linux
Simulator Platform	NS 2.28
Programming Languages	C++ and TCL
Optical Burst Switching Module	OBS 0.9a

We used the NS2 simulator (version 2.28) [41] as our simulation platform. The NS2 simulator has proved to be a reliable simulation platform for various technologies in the communications field [41]. The OBS-0.9a [42] module was used to simulate our OBS network. The OBS-0.9a module had the basic functions of an OBS network, which included burst assembly, shortest path routing, scheduling, and wavelength conversion.

To simulate the EDP scheme, we extended the functions of the OBS-0.9a module. We added a variable timer-based assembly scheme [32] for packets at the edge of the network, service class differentiation, and implemented deflection routing as a contention resolution scheme. The appendix provides details of the implementation.

4.2 Simulation Objectives

We set up our simulations to investigate the added benefits of the EDP scheme to the performance of an OBS network. Hence, we evaluated the performance of the EDP scheme in terms of throughput, goodput, end-to-end delay (T_{delay}), and burst loss P_{loss}.

The essence of the EDP scheme is to limit the number of the unnecessary burst deflections based on traffic requirements. The use of service class differentiation at the edge and core of the network contributes to discriminate between these traffic requirements. We aim to show that the use of service class differentiation in the form of the EDP scheme can improve the performance of deflection routing in an OBS network.

The objectives of our simulations are to determine whether:

1. The option of resolving contention with deflection routing increases throughput in an OBS network.

2. Deflection with the EDP scheme exhibits lower burst loss than deflection without the EDP scheme.

3. Deflection with the EDP scheme leads to higher throughput than deflection without the EDP scheme.

4. Deflection with the EDP scheme leads to lower total end-to-end delay of packets than deflection without the EDP scheme.

5. Deflection with the EDP scheme leads to higher goodput than deflection without the EDP scheme.

4.3 Network Topology

For our experiment, we used the network topology shown in Figure 4.1. The topology consists of 12 edges nodes and 6 core nodes. Each edge node can be viewed as the link to a metropolitan area network. Nodes are connected with DWDM fiber links, which transmit data optically. The nodal degree of the network topology is $N = 2.2$. The nodal degree indicates the level of connectivity between the nodes in the network. We define the nodal degree as, $N = \frac{2\ x\ \text{no. of bi-directional links}}{\text{no. of nodes}}$.

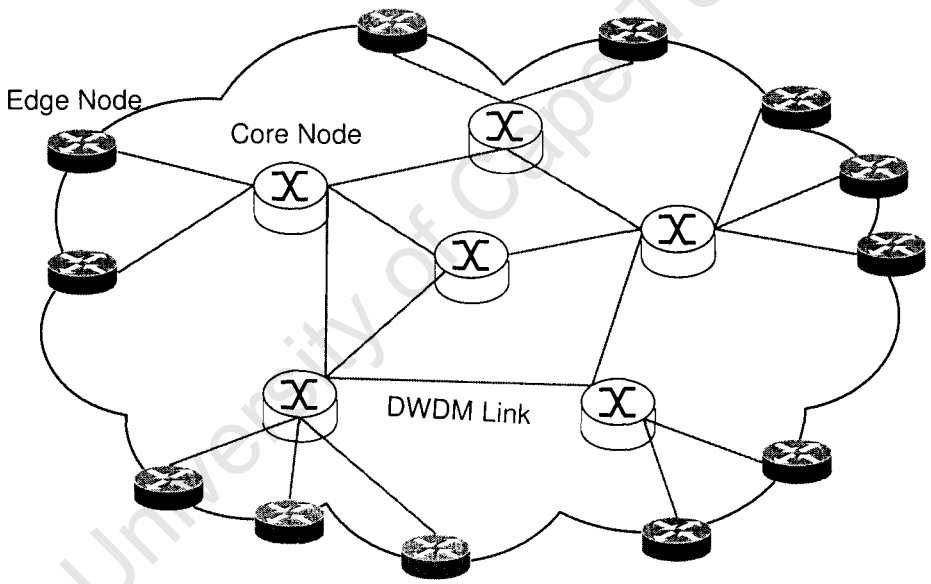


Figure 4.1 OBS Network of 12 Edge Nodes and 6 Core Nodes.

Our experiments were done under the following assumptions.

- The offset time is always large enough to prevent a burst from catching up with its corresponding burst header packet.
- There is no wavelength conversion and no optical buffering (FDLs) at all nodes in the network.

- The burst length depends on a variable period of assembly and a maximum burst length.
- The JET protocol is used to reserve network resources.
- The LAUC-VF algorithm is used to schedule bursts at all nodes.
- Djikstra's shortest path routing algorithm is used for routing bursts in the network.
- There is a uniform distribution of packets within each traffic class.
- There is a uniform distribution of the delay requirements of packets within each traffic class.
- The period of burst disassembly is negligible.

4.4 Simulation Parameters

In this section, we present the simulation parameters of our experiment. We show how internet traffic is generated at the edge nodes and describe its distribution within each traffic class.

4.4.1 Basic Setup

Table 4-2 illustrates the configurations of our OBS network. We limited the number of data channels and control channels to 1 in order to prevent the use of wavelength conversion. This limitation allowed us to strictly evaluate the effect of deflection routing in the OBS network.

The maximum burst length was 1 Megabyte (MB). This length is reached if and only if the burst queue is filled before the burst assembly period is over. Hence, a 1MB burst is a burst which was assembled based on the maximum burst length instead of a maximum period of burst assembly.

We set an offset time of 50 microseconds (μs) to ensure that burst payloads never reached their destination before their respective burst header packets. The link delay varied between 1 and 3 milliseconds (ms). This link delay corresponds to a distance ranging from 300 to 900 km between a node pair.

Table 4-2 Simulation Parameters of OBS Network simulation

Bandwidth per Channel	1 Gbps
Number of Data Channels	1
Number of Control Channels	1
Link Delay	1-3 ms
Offset Time	50 μs
Switching Time of Control Packet	1 μs
Packet Size	1000 bytes
Maximum Burst Length (L)	1MB

4.4.2 Traffic Generators

In order to model the burstiness and self-similarity of data traffic, we generated traffic according to a heavy-tailed Pareto distribution. Authors in [43] showed that multiplexing several heavy tailed Pareto distributions into the same queue can generate self-similar traffic. This distribution has the function $F(x) = 1 - \frac{1}{x^{\alpha}}$ where α is the shape parameter (tail index) that indicates the tail-heaviness of the distribution. Traffic of self-similar nature is characterised by the Hurst parameter, $H = \frac{(3-\alpha)}{2}$ where $1 < \alpha < 2$. In our experiment, $\alpha = 1.2$ and thus the Hurst parameter H = 0.9.

Incoming traffic is generated using ON-OFF sources as shown in Figure 4.2.

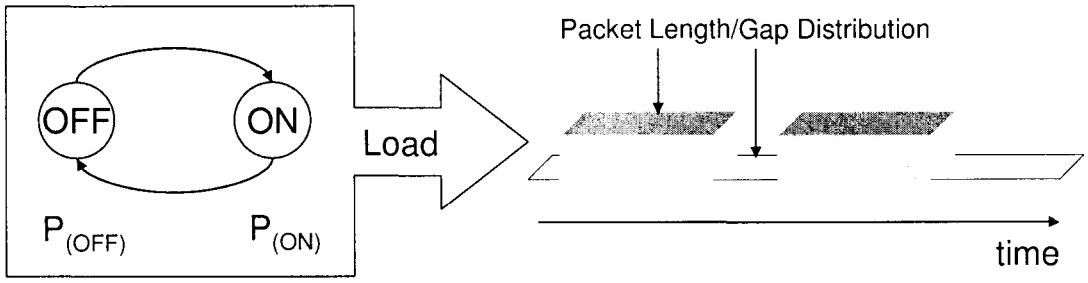


Figure 4.2 Traffic Source Generated using ON-OFF Periods.

The ON and OFF periods are Pareto distributed. The traffic load per source is the mean size of ON periods over the mean size of ON and OFF periods.

$$L_i = \frac{\overline{ON_i}}{\overline{ON_i} + \overline{OFF_i}}$$

The total load L is therefore the sum of loads Li generated by each source i. Hence, given N sources,

$$L = \sum_{i=1}^{N} L_i$$

4.4.3 Service Class Differentiation

Table 4-3 illustrates the distribution of different application types and the range of their delay requirements. Network control and interactive applications each represented 20% of the data traffic while responsive and timely applications made up the remaining traffic.

Table 4-3 Traffic Class Configuration for Burst Assembly

Traffic Class	Input Traffic Ratio	Delay Tolerance (ms)	Assembly Period (ms)	Emission Priority	Discard Priority
Network control	20%	50 – 70	45 - 85	3	1
Interactive	20%	80 – 100	75 - 105	2	0
Responsive	30%	110 – 130	105 - 135	1	2
Timely	30%	140 – 160	135 – 165	0	3

The emission and discard priorities of the data traffic varied between 0 and 3 (Table 4-3). Network control applications had the highest emission priority because they are necessary for the correct operation of the network. Although interactive applications have a high sensitivity to loss, they had the lowest discard priority. This low discard priority means that the application has stringent delay requirements and thus is not favorable to deflection. End-to-end delay (T_{delay}) is more important than loss (P_{loss}) in the case of interactive applications as late arrivals are unacceptable.

4.5 Simulation Cases

Our experiment investigated the cases of no-deflection, deflection and deflection with the EDP scheme.

Case 1 – No-deflection

In the case of no-deflection, the only method of contention resolution is the drop policy. Hence, if a burst cannot be scheduled, it is dropped.

Case 2 – Deflection

In the case of deflection, we implemented a deflection routing algorithm of order 1. In other words, deflection is only attempted on the next shortest alternative hop to destination. However, there is no limit to the possible number of deflections on a burst until it reaches its destination. If a burst cannot be scheduled on the next alternative hop, it is dropped.

Case 3 – Deflection with EDP scheme

In the case of deflection with the EDP scheme, we also implemented a deflection routing algorithm of order 1. However, the number of burst deflections is the limited, based on the emission and discard priority of the burst. If a burst cannot be scheduled on the next alternative hop, it is dropped.

In the next chapter, we present and analyse the results obtained from simulating the cases of no-deflection, deflection and deflection with the EDP scheme.

5 Experimental Results and Analysis

This chapter presents an evaluation of the performance of deflecting bursts with the EDP scheme using the NS2 framework. The objective of our work is to make deflection routing a suitable candidate for supporting wavelength conversion as a method of contention resolution in OBS. To achieve our objective, we aim at reducing the late arrivals of delay intolerant packets at their destination due to deflection. Burst deflection would therefore have a lesser effect on packet latency in the network.

The key QoS metric in our experiment is goodput. However, goodput is dependent on other intermediate QoS metrics. Improving the goodput of deflection routing implies a high throughput and a low number of late packet arrivals. Figure 5.1 shows the dependencies that link these intermediate QoS metrics to goodput. The highlighted boxes represent the QoS metrics used to evaluate the deflection with EDP scheme in the subsequent sections of this chapter.

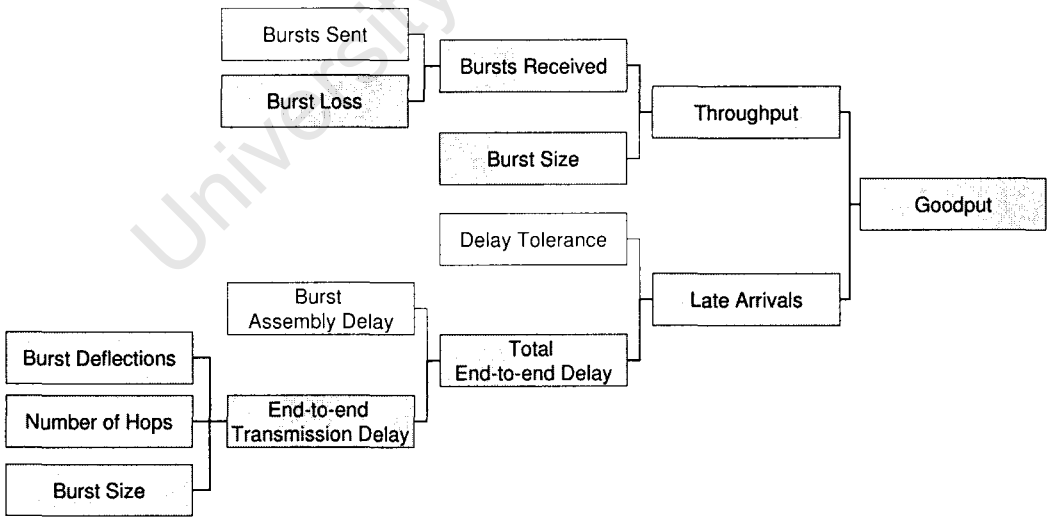


Figure 5.1 Dependency Graph of the Performance Metrics in our OBS Network.

This chapter is organised as follows. Section 5.1 demonstrates that our simulator exhibits the correct and predictable behaviour. Section 5.2 analyses the results obtained from measuring the throughput-related performance metrics (Figure 5.1). In Section 5.3, we examine the results obtained from measuring the delay-related performance metrics (Figure 5.1). Section 5.4 analyses the results of the overall goodput measured during our experiment. Section 5.5 summarizes the analysis of our results and Section 5.6 describes the limitations of our experiment.

5.1 System Validation

To verify the predictability of our network simulator, we collected data that would help us ensure the correct behavior of the simulator. Our network consisted of 12 edge nodes and 6 core nodes. Only the edge nodes could generate traffic. The maximum data rate of each edge node was 1Gb, and thus the maximum sending rate of the entire network was 12Gb/s. The traffic load L is the ratio of data sent per link and the maximum data rate per link. Therefore, when the load $L=1$ we expect the data sent to approach 12Gb for a simulation time of 1 second. We also expect the data sent to be proportional to the traffic load.

As the load increases, we expect a higher loss of data in the network. The increase in data loss is due to a higher load, which implies that a high number of bursts are competing for network resources. As a result, the availability of resources in the OBS network is reduced. Figure 5.2 illustrates the collection of data sent and data received versus traffic load.

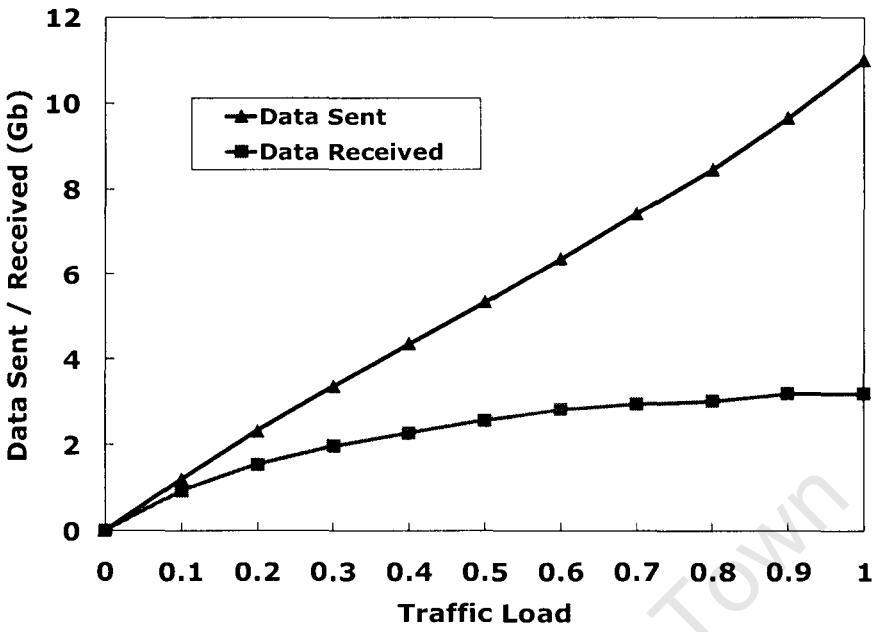


Figure 5.2 Data Sent / Received versus Traffic Load.

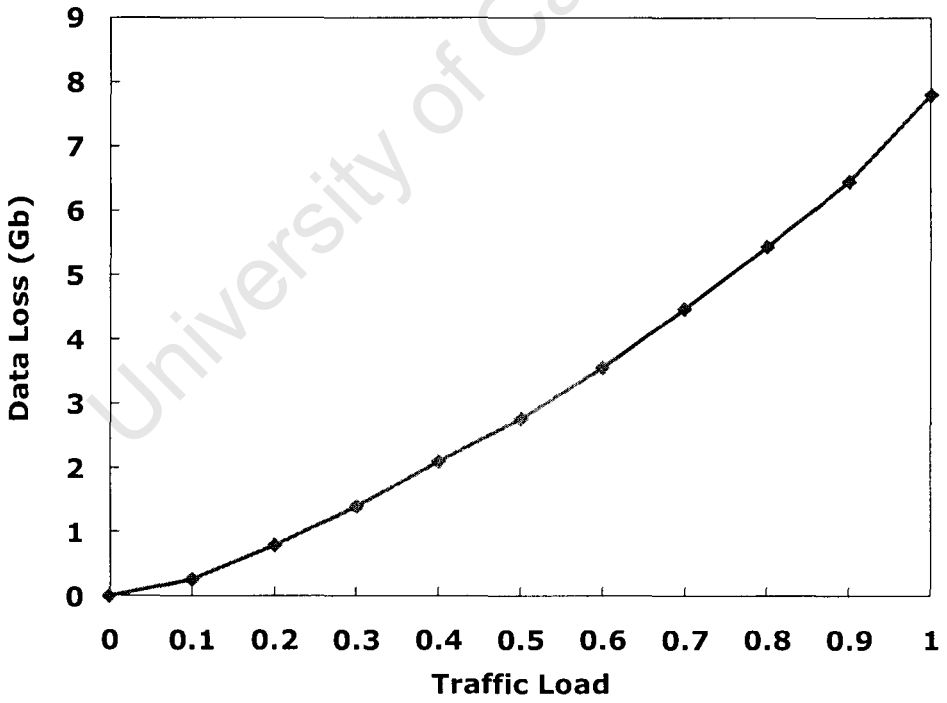


Figure 5.3 Data Loss versus Traffic Load.

In Figure 5.2, we observe that the data sent increases as the load increases. As the load approaches $L=1$, the data sent approaches 12Gb as predicted. We also see that the plot of the data sent against the traffic load is linear. Hence, as expected the amount of data sent is proportional to the traffic load. In addition, Figure 5.2 shows that the amount of data received increases as the load increases. However, when the load $L \geq 0.7$, the amount of data received becomes saturated even though the data sent keeps increasing. This saturation condition indicates that the availability of network resources decreases rapidly when the traffic load is high. Hence, we observe that as the load increases, the space between the plot of data sent and the plot of the data received expands. This expanding space represents the expected increase in data loss. Figure 5.3 shows the plot of data loss against the traffic load. The amount of data loss is the difference between the amount of data sent and the amount of data received (Figure 5.2).

5.2 Throughput Analysis

In this section, we evaluate the performance of the EDP scheme using the QoS metrics that are linked to throughput (Figure 5.1). We then analyse the results obtained from measuring the overall throughput. For the remainder of this chapter, we will refer to the no-deflection scheme as case 1, to the deflection scheme as case 2 and to the deflection with EDP scheme as case 3.

Figure 5.4 shows the burst loss probability versus traffic load for case 1, 2 and 3. In the simulation environment, the probability of losing bursts is the ratio between the number of bursts lost and the number of bursts sent in the network.

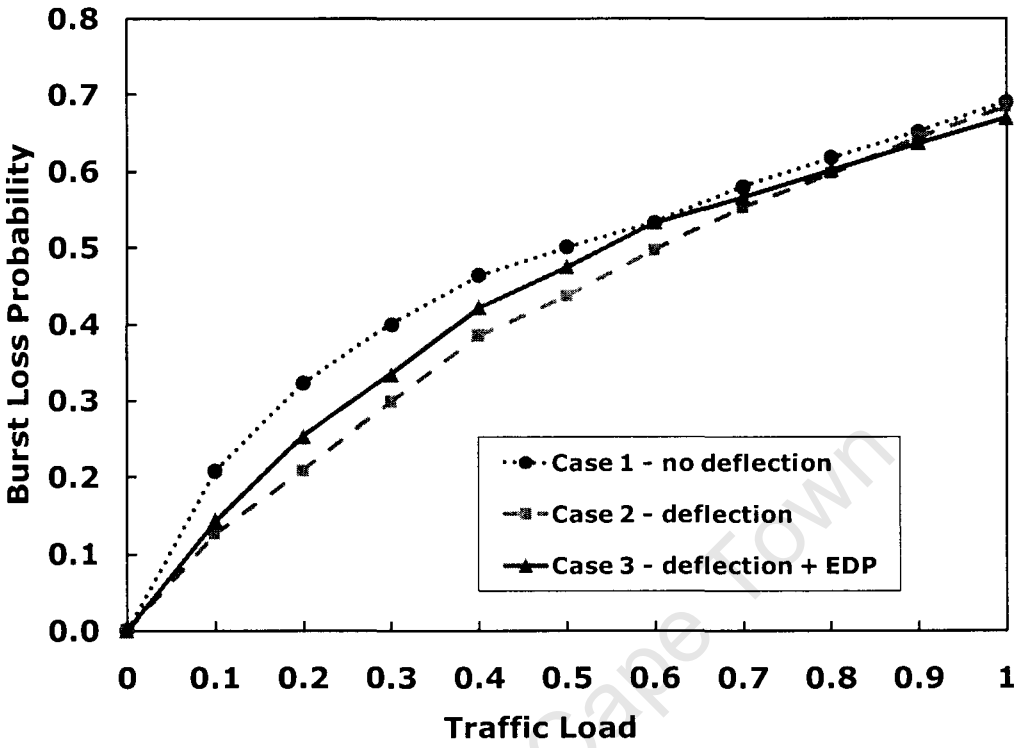


Figure 5.4 Burst Loss Probability versus Traffic Load

We observe that at low loads, case 2 performs better than case 1 and case 3. Case 2 performs better at low loads because all bursts can suffer an unlimited number of deflections and the availability of network resources is high. Hence, the majority of deflected bursts eventually reach their destination. However, we notice that at loads $L >= 0.8$, case 3 has lower burst loss than case 1 and case 2. As the load increases, the number of bursts generated increases and therefore the probability of contention increases. Attempting too many burst deflections further increases the probability of contention when the traffic load is high. Hence, we see that case 2 has a negative effect on burst loss at high loads. On the other hand, case 3 deflects bursts selectively and therefore avoids this negative effect at high traffic loads. The difference between case 2 and case 3 for $L >= 0.8$ is of the order 10^{-2} in terms of burst loss probability. This difference represents approximately 20MB, which is significant.

Figure 5.5 illustrates the average size of bursts received versus the traffic load for each contention resolution scheme.

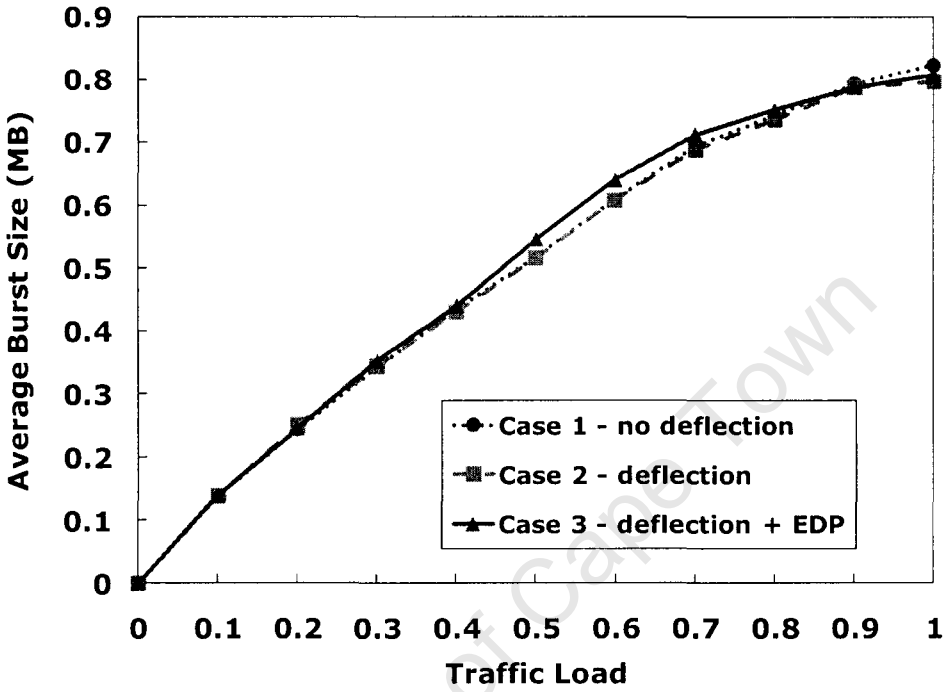


Figure 5.5 Average Size of Bursts Received versus Traffic Load.

We notice that the average size of bursts increases with increasing traffic load. As the traffic load becomes higher, the rate at which bursts are assembled increases. Thus, the burst size will be larger for a high rate of assembly than for a low rate of assembly. The maximum burst size is 1MB and we can see that the average burst size is always less than the maximum burst size. When $0.4 \leq L \leq 0.9$, we observe that the average burst size of case 3 is the highest. This higher average burst size is because the EDP scheme in case 3 uses delay tolerance as a criterion to discriminate between bursts. Delay tolerant bursts are more favored for deflection than delay intolerant bursts. Therefore, the majority of deflected bursts that reach their destination in case 3 will be delay tolerant. In addition, bursts that are delay tolerant have the longest burst assembly delay. This extended burst assembly period

means that delay tolerant bursts have the largest size. As a result, the large burst size of delay tolerant bursts will increase the average size of bursts received in case 3.

Figure 5.6 plots the number of bursts received against the traffic load for each contention resolution scheme.

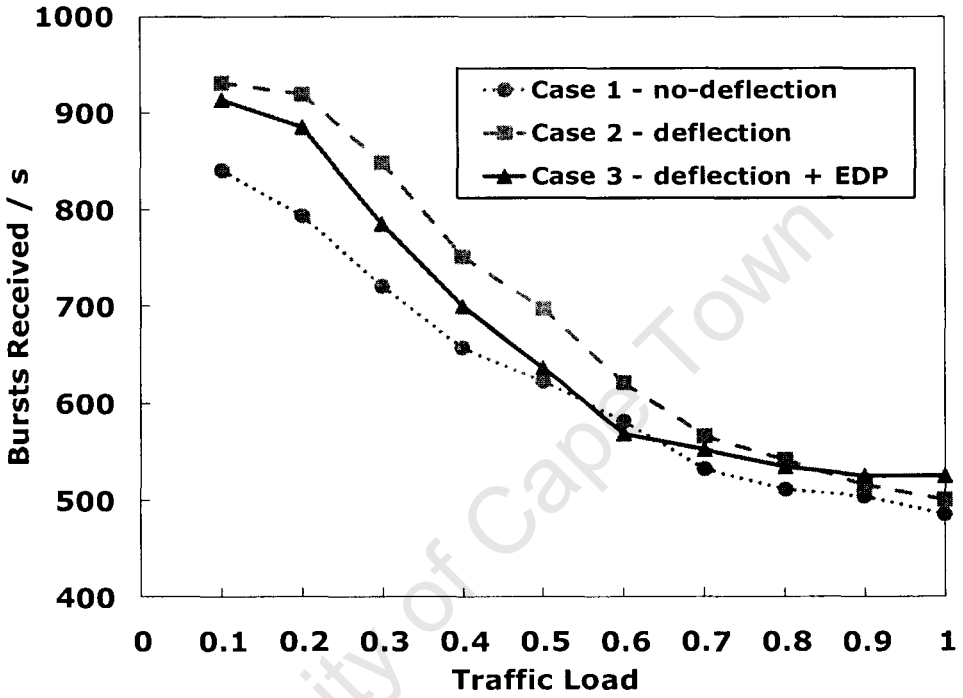


Figure 5.6 Bursts Received per second versus Traffic Load

We observe that the number of bursts received decreases as the load increases in all cases. We would expect the number of bursts received to increase as the load increased. However, we saw in Figure 5.5 that the average burst size increased as the traffic load increased. Large bursts are more susceptible to contention because they require more scheduling resources. Hence, the number of bursts received decreases as the load increases. Still, we would expect the throughput to increase because of the large size of bursts received (Figure 5.5). At very high loads ($L > 0.85$), we see that case 3 performs better than case 2 because it implements selective burst deflections. Thus, case 3 avoids the negative effect of the unlimited number of possible burst deflections present in case 2.

Figure 5.7 plots throughput versus the traffic load for each contention resolution scheme.

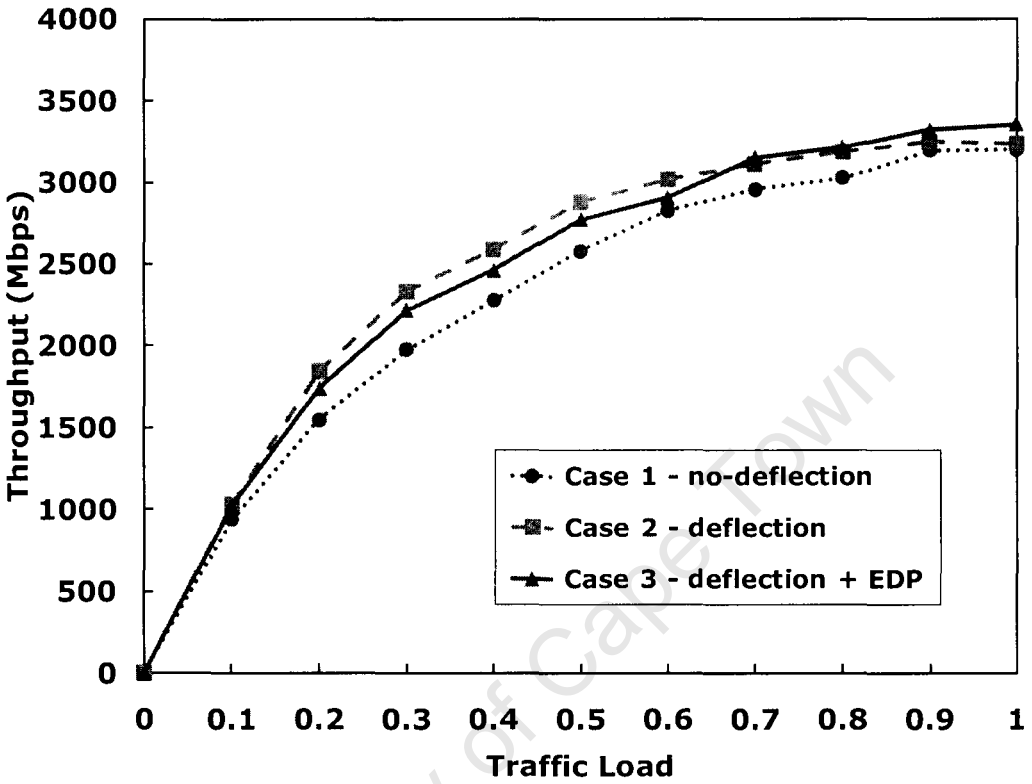


Figure 5.7 Throughput versus Traffic Load.

As expected, we see that the throughput increases with increasing traffic load. We also notice that case 2 performs better than case 1 and case 3 for loads $L < 0.7$. However, case 3 has the highest throughput when $L \geq 0.7$. Throughput can be seen as the product of the number of bursts received (Figure 5.6) and the average size of bursts received (Figure 5.5). Hence, the better performance of case 3 at high loads in terms of average size of bursts received, and at very high loads in terms of number of bursts received results in a higher throughput than case 2 when $L \geq 0.7$.

5.3 Delay Analysis

In this section, we measure the performance of the no-deflection scheme (case 1), the deflection scheme (case 2), and the deflection with EDP scheme (case 3) with

the intermediate QoS metrics that are linked to late arrivals (Figure 5.1). We then analyse the results of measuring the proportion of late packet arrivals.

Figure 5.8 plots the number of burst deflections per second versus traffic load for the deflection scheme (case 2) and the deflection with EDP scheme (case 3).

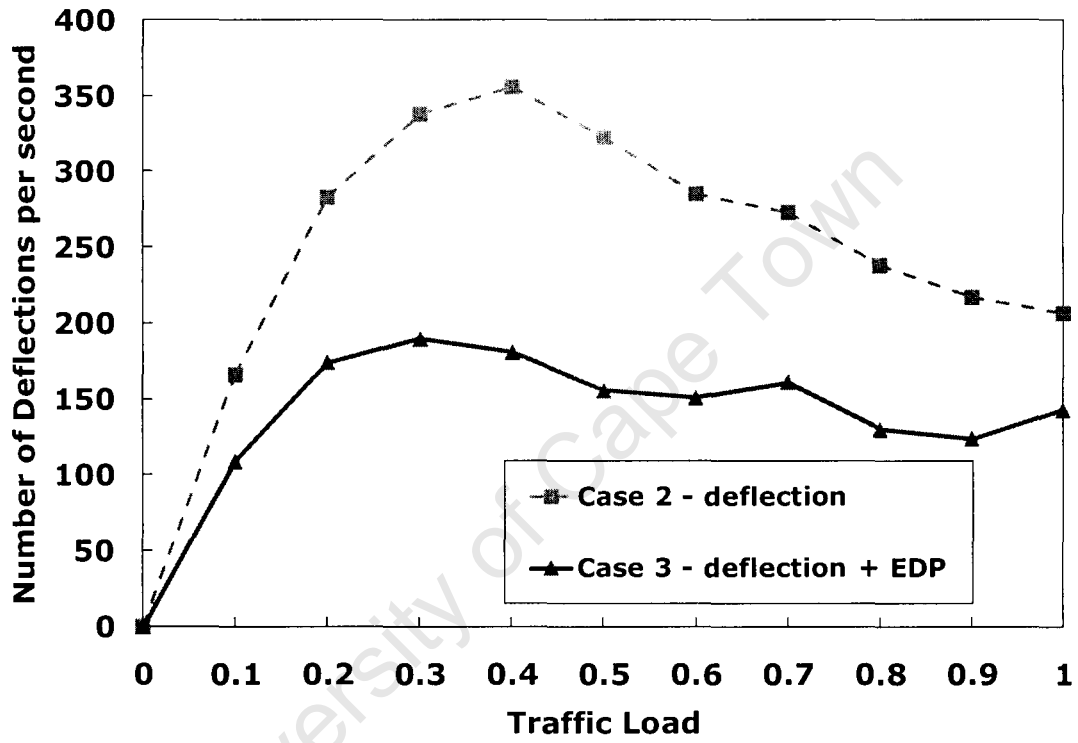


Figure 5.8 Number of Deflections per second versus Traffic Load.

We notice that in case 2 and case 3, the number of deflections increases when $0 \le L \le 0.4$, and then decreases when $L > 0.4$. This increase in the number of burst deflections for $0 \le L \le 0.4$ is due to the increasing number of bursts contending for resources. Deflection routing provides extra scheduling resources and, therefore the number of burst deflections increases as the load increases. However, when the load $L > 0.4$, the number of deflections decreases because of the large burst sizes (Figure 5.5) and the high traffic load. Hence, scheduling becomes difficult even when using deflection routing to resolve contention. We also observe that case 3 has a lower

number of deflections than case 2 at all traffic loads. The discard priority limits the number of possible burst deflections. If the discard priority of a burst equals 0 when a deflection is needed, the burst is dropped. As a result, case 3 carries out less burst deflections than case 2.

Figure 5.9 shows the average number of burst hops per second versus traffic load for each contention resolution scheme.

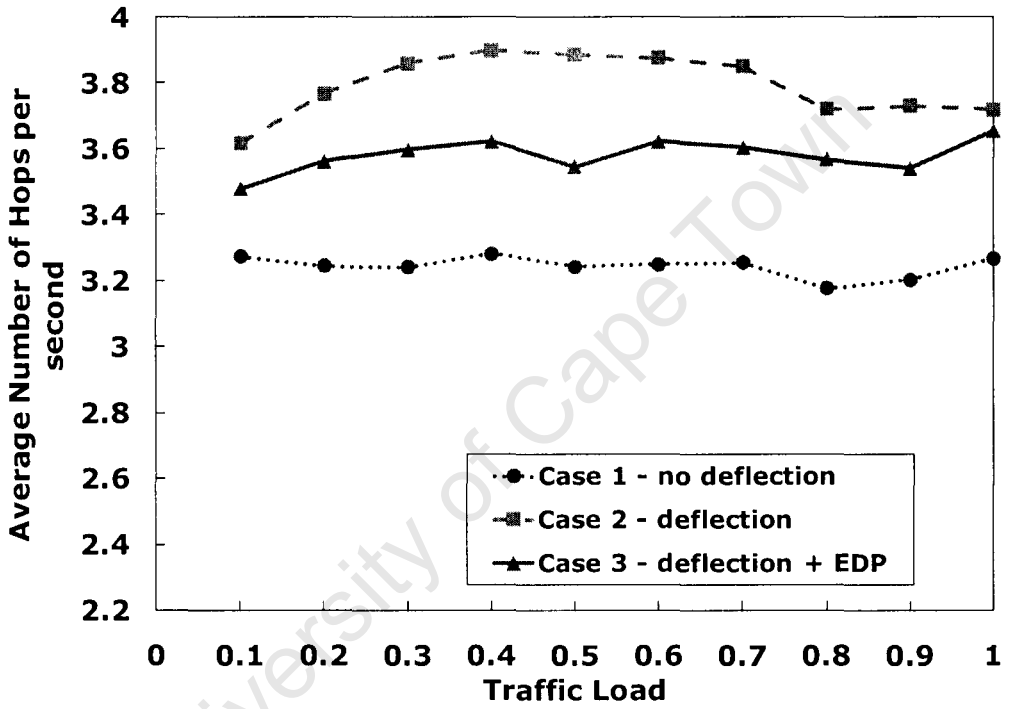


Figure 5.9 Average Number of Hops per second versus Traffic Load.

We observe that case 1 has the lowest average number of hops at all loads. Conversely, case 2 has the highest average number of hops at all loads. Hence, a high number of deflections increases the number of hops needed to reach destination. We note that case 3 has a higher average number of hops than case 1 and a lower average number of hops than case 2.

Figure 5.10 plots the average burst end-to-end transmission delay versus traffic load. The end-to-end transmission delay is the time elapsed during the

transmission of a burst from source to destination. Hence, this delay does not include the burst assembly delay.

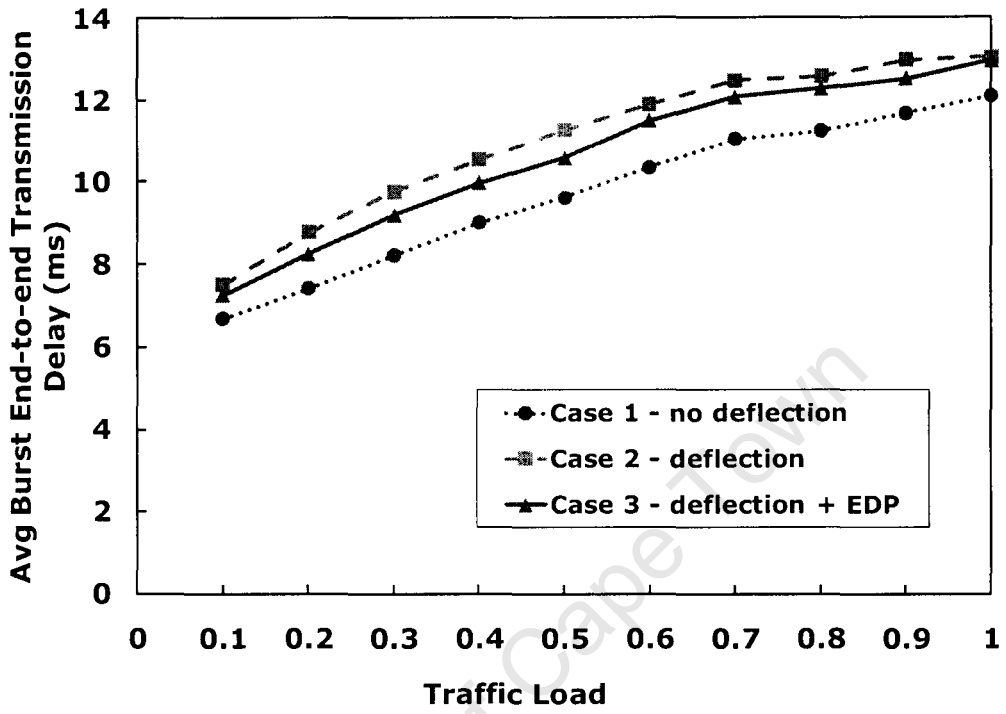


Figure 5.10 Average Burst End-to-end Transmission Delay versus Traffic Load.

We see that case 2 has a higher end-to-end transmission delay than case 3 at all loads. This higher end-to-end transmission delay is because case 2 has the highest average number of hops, as seen as in Figure 5.9. Thus, a high number of deflections implies a high end-to-end transmission delay. Figure 5.10 also shows longer end-to-end transmission delays as the load becomes higher. This increase in end-to-end transmission delays is due to the increasing size of bursts at high loads (Figure 5.5). The transmission time of bursts from the edge node depends on the link capacity and the burst size. Hence, an edge node will take a longer period to transmit a large burst than to transmit a small burst.

Figure 5.11 plots the average total end-to-end delay versus traffic load for each contention resolution technique. The total end-to-end delay is the sum of the burst assembly time and the end-to-end transmission delay. We observe that case 3

has a higher total end-to-end delay than case 1 and case 2 when $0.3 \leq L \leq 0.8$. This higher delay is due to the large proportion of delay tolerant bursts received when deflecting with the EDP scheme. These delay tolerant bursts are more susceptible to deflection than other bursts and hence the delay they incur in the network increases the average total end-to-end delay. We also notice a sharp decline in the total end-to-end delay for all three contention resolution schemes when $L > 0.6$. This decline in total end-to-end delay is due to the high rate of burst assembly at high loads. Hence, the maximum burst size is reached before the end of the assembly period. As a result, bursts that are sent before the end of their assembly period will have a lower total end-to-end delay.

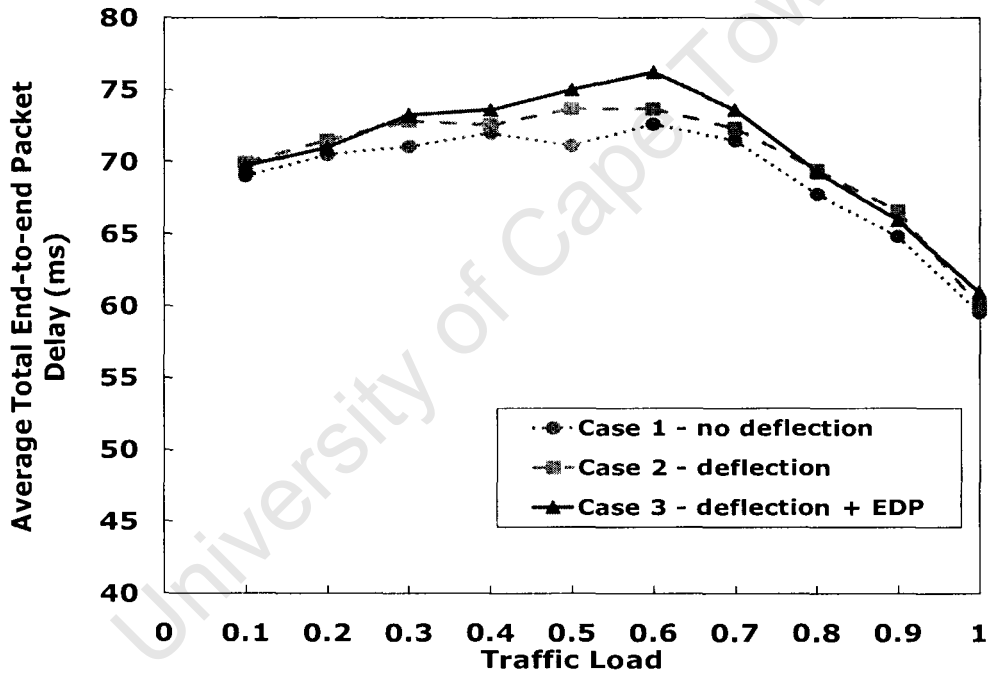


Figure 5.11 Average Total End-to-end Delay versus Traffic Load.

Figure 5.12 plots the percentage of late packet arrivals versus traffic load for each contention resolution scheme.

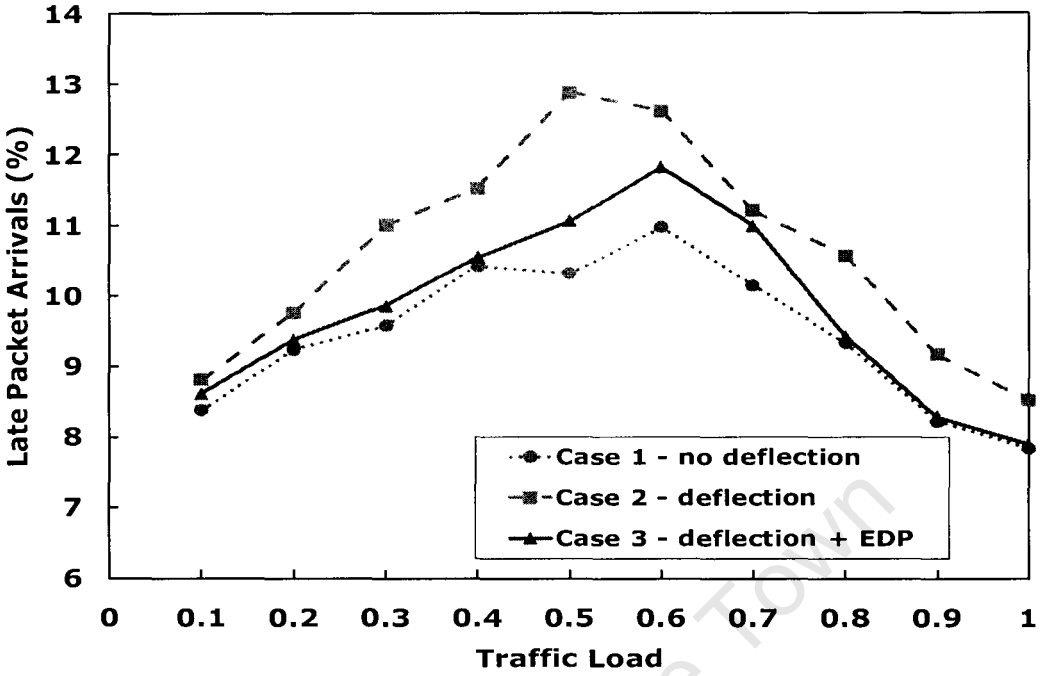


Figure 5.12 Late Packet Arrivals versus Traffic Load.

We observe that case 1 has the lowest percentage of late packet arrivals because it does not implement deflection routing. Hence, bursts do not incur any extra delay due to deflection routing. When $L = 0.5$, case 2 has the worst performance with up to 13% of packets arriving late, which is 2% higher than in case 3. This difference of 2% in late packet arrivals represents approximately 7000 packets, which is significant. We notice that case 3 performs better than case 2 at all loads. In contrast to case 2, case 3 performs selective burst deflections, and therefore minimizes the number of late packet arrivals.

5.4 Goodput Analysis

In this section, we complete our analysis with the measurement of Goodput for the no-deflection scheme (case 1), the deflection scheme (case 2) and the deflection with EDP scheme (case 3). In addition, the efficiency of the deflection scheme (case 2) and the deflection with EDP scheme is measured in terms of the amount of goodput per deflection.

Figure 5.13 plots goodput versus traffic load for each contention resolution scheme.

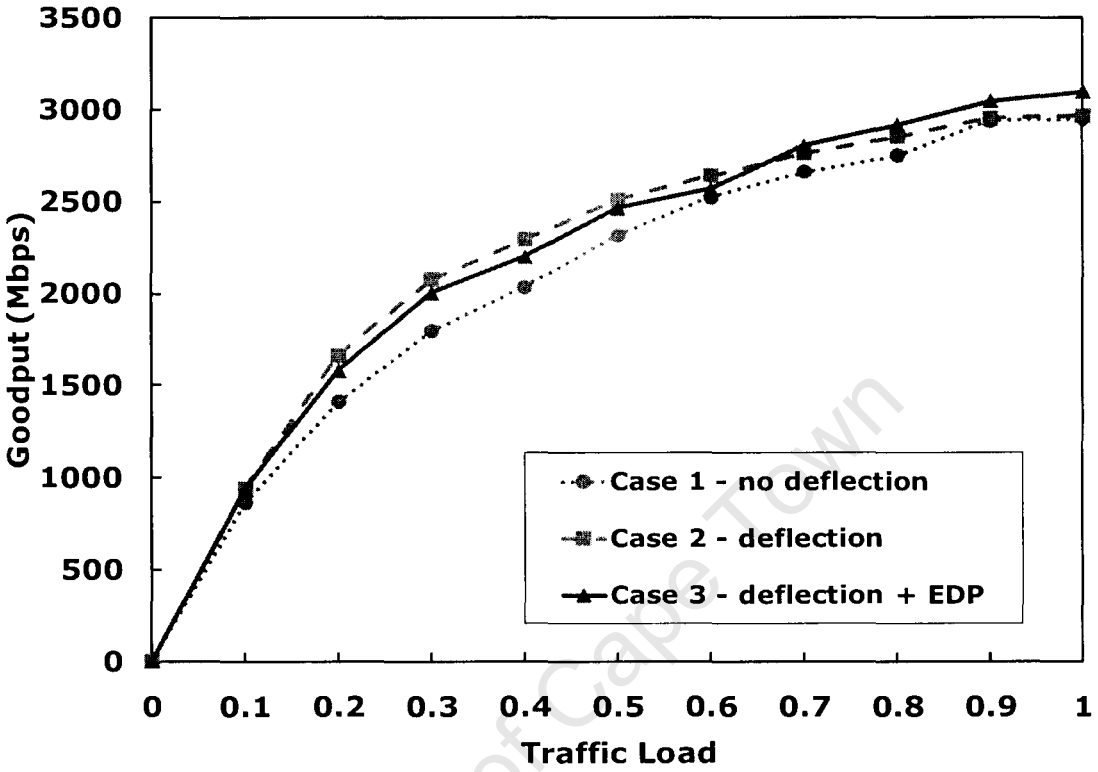


Figure 5.13 Goodput versus Traffic Load.

Goodput represents the amount of data that reaches destination within its delay requirements. Hence, as seen in Figure 5.1, goodput depends on throughput and the proportion of late packet arrivals. In Figure 5.13, we notice a higher goodput in case 2 than in case 3 when $L < 0.7$. However, at high loads ($L \geq 0.7$) case 3 has a higher goodput than case 2. Case 3 performs better than case 2 because it has a higher throughput when $L \geq 0.7$ (Figure 5.7) and a lower proportion of late packet arrivals (Figure 5.12). In addition, we see that the difference in goodput between case 2 and case 3 is minimal at low loads. It is important to note that although case 1 performs better than case 2 and case 3 in terms of late packet arrivals, it has a lower goodput than case 2 and case 3 at all loads. Case 1 has the lowest goodput because it has a lower throughput than case 2 and case 3 (Figure 5.7).

Figure 5.14 shows the amount of goodput per deflection versus traffic load for the case of the deflection scheme and the deflection with EDP scheme.

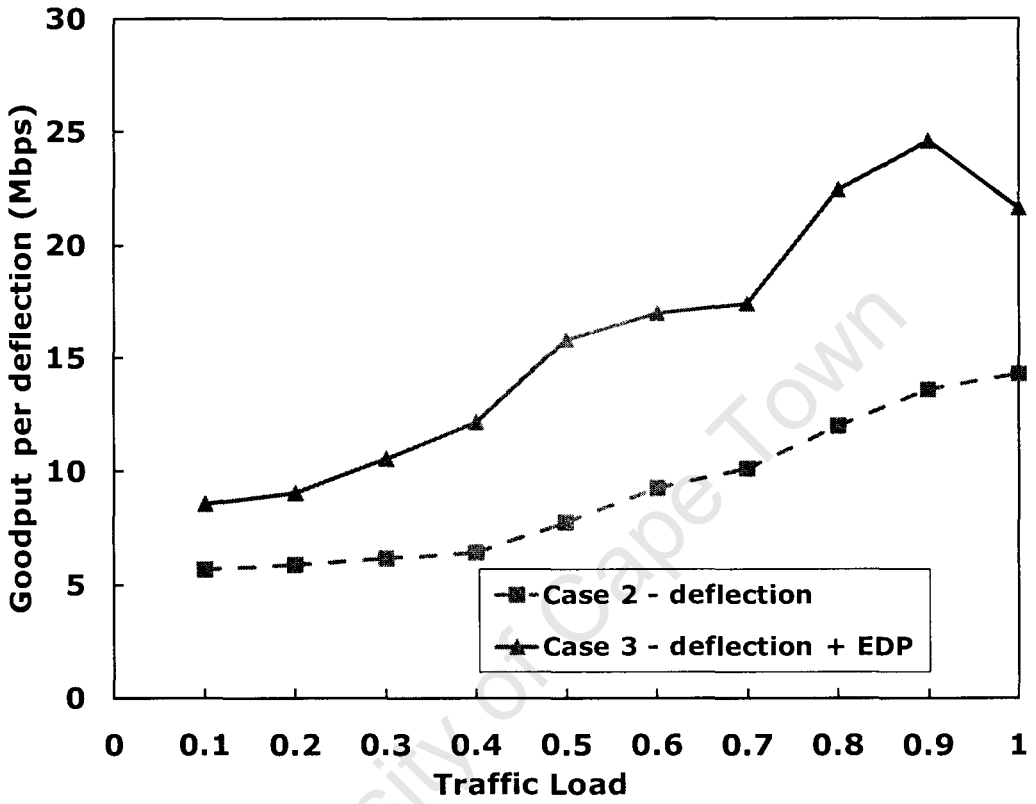


Figure 5.14 Goodput per Deflection versus Traffic Load

We observe that case 3 has a higher goodput per deflection ratio than case 2 at all loads. As the load becomes higher, the margin between case 3 and case 2 increases. As the load increases, the probability of burst contentions increases. Hence, it is important to selectively deflect bursts in an efficient manner at high loads. Case 3 applies the EDP scheme to perform selective burst deflections, and therefore performs better than case 2. We also notice a sharp increase in the amount of goodput per deflection in case 3 when $0.7 \leq L \leq 0.9$. This sharp increase corresponds to the better performance of case 3 relative to case 2 in terms of goodput (Figure 5.13).

5.5 Concluding Remarks

In this chapter, we analyzed the performance of the deflection with EDP scheme with the QoS metrics shown in Figure 5.1. In this analysis, we compared the performance of the no-deflection scheme, the deflection scheme and the deflection with EDP scheme.

We verified the fact that deflection provides higher throughput at the cost of higher end-to-end transmission delay. We learnt that the deflection with EDP scheme has lower throughput than the deflection scheme for loads $L < 0.7$, but performs better than the deflection scheme for loads $L \geq 0.7$. We also noted that the average end-to-end transmission delay is one of the key factors that affect the proportion of late packet arrivals. Indeed, the deflection with EDP scheme, which has a lower end-to-end transmission delay than the deflection scheme, also has a lower proportion of late packet arrivals than the deflection scheme.

In terms of goodput, the deflection scheme outperforms the deflection with EDP scheme for loads $L < 0.7$, while the deflection with EDP scheme outperforms the deflection scheme for loads $L \geq 0.7$. Although the deflection scheme had higher goodput than the deflection with EDP scheme at low loads, the deflection with EDP scheme carried out fewer burst deflections, with a minimal loss in goodput. The efficiency of the deflection with EDP scheme was highlighted in Figure 5.14. Figure 5.14 showed that the deflection with EDP scheme has a higher goodput per deflection ratio, and is therefore more efficient than the deflection scheme.

5.6 Limitations

We did not investigate the out-of-order arrivals of packets at their destination. Out-of-order arrivals are considered a limitation of deflection routing. However, this limitation is secondary to the problem of late packet arrivals that deflection routing causes.

6 Conclusions and Future Work

In this chapter, we highlight our contributions (Section 6.1) and make recommendations for future work (Section 6.2).

6.1 Conclusions

To improve the QoS in OBS networks, our study emphasized the need to resolve contention in an efficient and cost-effective manner. We therefore proposed to combine an emission and discard priority (EDP) scheme with deflection routing to better meet the performance requirements of traffic in OBS networks. The proposed deflection with EDP scheme consisted in assigning emission and discard priorities to bursts in order to deflect them selectively in the core network. The objective of the deflection with EDP scheme was to reduce the proportion of late packet arrivals caused by deflection routing while maintaining its high network throughput. In this way, the efficiency of the deflection routing and the network goodput would be improved.

During our experiment, we compared the performance of the no-deflection scheme, the deflection scheme and the deflection with EDP scheme. Based on our results, we showed that the deflection with EDP scheme outperforms the deflection scheme in terms of end-to-end transmission delays and proportion of late packet arrivals. Furthermore, we showed that the deflection with EDP scheme has higher goodput than the deflection scheme at high loads ($L \geq 0.7$). Although the deflection scheme has higher goodput than the deflection with EDP scheme for loads $L < 0.7$, we found that the deflection with EDP scheme has greater efficiency than the deflection scheme in terms of goodput per deflection.

In brief, the proposed EDP scheme enhances the efficiency of deflection routing with selective burst deflections. This higher efficiency is more apparent at high loads where the performance of deflection routing is reduced in terms of

goodput. In addition, the EDP scheme helps to reduce the proportion of late packet arrivals caused by deflection routing and enhances the goodput of deflection routing for loads $L \geq 0.7$. Further studies concerning the out-of-sequence arrivals of packets are needed to determine the effectiveness of deflection routing as a method of contention resolution for OBS networks.

6.2 Future Work

Our work can be extended in the following directions.

- Investigate the effect of deflecting bursts with the EDP scheme on network topologies of various nodal degrees.
- Investigate the impact of the EDP scheme on out-of-order packet arrivals caused by deflection routing in OBS networks.
- Determine whether the combination of the EDP scheme with wavelength conversion and deflection routing could further improve QoS in OBS networks in an efficient and cost-effective manner.
- Analyse the effects of the EDP scheme on other contention resolution methods such as wavelength conversion, burst segmentation and optical buffering.

References

[1] W. Willinger, M. S. Taqqu, R. Sherman, and D. V. Wilson, "Self-similarity through high-variability: statistical analysis of Ethernet LAN traffic at the source level," *IEEE ACM Transactions on Networking,* vol. 5, pp. 71-86, 1997.

[2] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson, "On the Self-Similar Nature of Ethernet Traffic (Extended Version)," *IEEE/ACM Transactions on Networking,* vol. 2, pp. 1-15, 1994.

[3] J. Dhliwayo, "Developing a fiber optic backbone for Africa," http://www.corning.com/docs/corporate/discovery_center/innovation_library/2004/NTR100763.pdf, August 2007.

[4] MatisseNetworks, "EtherBurst Optical Switch (Technology and Architecture)," http://www.matissenetworks.com/downloads/EtherBurst%20Technology%20and%20Architecture.pdf, September 2007.

[5] C.-F. Hsu, T.-L. Liu, and N.-F. Huang, "Performance analysis of deflection routing in optical burst-switched networks," in *Proceedings IEEE INFOCOM*, 2002, pp. 66-73.

[6] C. Qiao and M. Yoo, "Optical burst switching (OBS) - a new paradigm for an Optical Internet," *Journal of High Speed Networks,* vol. 8, pp. 69 - 84, 1999.

[7] M. Yoo and C. Qiao, "Just-enough-time(JET): a high speed protocol for bursty traffic in optical networks," in *SPIE Proceedings, All Optical Communication Systems: Architecture, Control and Network Issues.* vol. 3230, 1997, pp. 79--90.

[8] J. Wei and R. McFarland, "Just-In-Time Signaling for WDM Optical Burst Switching Networks," *IEEE Journal of Lightwave Technology,* vol. 18, pp. 2019-2037, December 2000.

[9] J. P. Jue and V. Vokkarane, *Optical Burst Switched Networks*, 1 ed. vol. 1: Springer, 2004.

[10] H. Buchta, "Analysis of Physical Constraints in an Optical Burst Switched Network." vol. PhD: Technische Universität Berlin, 2005, p. 200.

[11] Y. Xiong, M. Vandenhoute, and H. Cankaya, "Control Architecture in Optical Burst Switched WDM Networks," *IEEE Journal on Selected Areas in communications,* vol. 18, pp. 1838-1851, October 2000.

[12] X. Yu, Y. Chen, and C. Qiao, "Study of traffic statistics of assembled burst traffic in optical burst switched networks.," in *Proceedings of Opticomm*, 2002, pp. 149-159.

[13] A. Ge, F. Callegati, and L. Tamil, "On Optical Burst Switching and Self Similar Traffic," *IEEE Communication Letters*, vol. 4, pp. 98-100, March 2000.

[14] X. Cao, Y. C. J. Li, and C. Qiao, "Assembling TCP/IP Packets in Optical Burst Switched Networks," in *Proceedings of IEEE Globecom*, Taiwan, 2002, pp. 2808-2812.

[15] C. Yang, Q. Chunming, and Y. Xiang, "Optical burst switching: a new area in optical networking research," *IEEE Network*, vol. 18, pp. 16-23, 2004.

[16] M. Izal and J. Aracil, "On the Influence of Self-similarity on Optical Burst Switching Traffic," in *Proceedings of IEEE Globecom*, 2002, pp. 2308- 2312.

[17] K. Dolzer and C. Gauger, "On Burst Assembly in Optical Burst Switching Networks - A Performance Evaluation of Just-Enough-Time," in *Proceedings of the 17th International Teletraffic Congress*, Salvador da Bahia, Brazil, 2001, pp. 149-160.

[18] I. Widjaja, "Performance analysis of burst admission-control protocols," in *Proceedings of IEE Communications*, 1995, pp. 7-14.

[19] J. Turner, "Terabit Burst Switching," *Journal of High Speed Networks,* vol. 8, pp. 3--16, 1999.

[20] J. Teng, "A study of optical burst switched networks with the jumpstart just-in-time signaling protocol." vol. PhD Raleigh: North Carolina State University, 2004.

[21] K. G. Dolzer, C.M. Späth, J. Bodamer, S., "Evaluation of reservation mechanisms for optical burst switching," *AEÜ International Journal of Electronics and Communications,* vol. 55, pp. 18-26, 2001.

[22] J. Xu, C. Qiao, J. Li, and G. Xu, "Efficient Channel Scheduling Algorithms in Optical Burst Switched Networks," in *Proceedings of IEEE Infocom*, 2003, pp. 2268-2278.

[23] A. S. Acampora and I. A. Shah, "Multihop lightwave networks: A comparison of store-and-forward and hot-potato routing," *IEEE Transactions on Communications,* vol. 40, pp. 1082–1090, January 1992.

[24] F. Forghieri, A. Bononi, and P. R. Prucnal, "Analysis and comparison of hotpotato and single-buffer deflection routing in very high bit rate optical mesh networks," *IEEE Transactions on Communications,* vol. 43, pp. 88–98, January 1995.

[25] A. Bononi, G. A. Castanon, and O. K. Tonguz, "Analysis of hot-potato optical networks with wavelength conversion," *IEEE/OSA Journal of Lightwave Technology,* vol. 17, pp. 525–534, April 1999.

[26] V. Vokkarane, J. P. Jue, and S. Sitaraman, "Burst segmentation: An approach for reducing packet loss in optical burst switched networks," in *Proceedings of IEEE ICC*, 2002, pp. 2673-2677.

[27] V. Vokkarane and J. Jue, "Prioritized Burst Segmentation and Composite Burst Assembly Techniques for QoS Support in Optical Burst-Switched Networks," *IEEE Journal on Selected Areas in Communications (JSAC),* vol. 21, pp. 1198-1209, September 2003.

[28] K. Bharrathsingh, "Quality of experience as an integral part of network engineering," in *Nortel Technical Journal*, 2005.

[29] R. Braden, D. Clark, and S. Shenker, "Integrated Services in the Internet Architecture: an Overview," Memo 1633, 1994.

[30] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, *An Architecture for Differentiated Service*: RFC Editor, 1998.

[31] M. Yoo, C. Qiao, and S. Dixit, "QoS Performance of Optical Burst Switching in IP-Over-WDM Networks," *Selected Areas in Communications, IEEE Journal on,* vol. 18, pp. 2062-2071, October 2000.

[32] P. Acquaah and H. A. Chan, "A Variable Timer-based Assembly Algorithm for Optical Burst Switched Networks," in *Proceedings of International Conference on Telecommunications (ICT)*, 2006.

[33] M. Klinkowski, D. Careglio, S. Spadaro, and J. Sole-Pareta, "Impact of burst length differentiation on QoS performance in OBS networks," in *Proceedings of International Conference on Transparent Optical Networks*, 2005, pp. 91-94.

[34] H. C. Cankaya, S. Charcranoon, and T. S. El-Bawab, " A preemptive scheduling technique for OBS networks with service differentiation," in *Proceedings of IEEE GLOBECOM* 2003, pp. 2704-2708.

[35] Z. Qiong, V. M. Vokkarane, J. P. Jue, and C. Biao, "Absolute QoS Differentiation in Optical Burst-Switched Networks," *IEEE Journal on Selected Areas in Communications (JSAC) - Optical Communications and Networking Series,* vol. 22, pp. 1781-1795, 2004.

[36] Y. Chen, M. Hamdi, and D. H. K. Tsang, "Proportional QoS over OBS networks," in *Proceedings of IEEE Globecom*, 2001, pp. 1510-1514.

[37] F. Poppe, K. Laevens, H. Michiel, and S. Molenaar, "Quality-of-service differentiation and fairness in optical burst-switched networks," in *Proceedings of SPIE OptiComm*, 2002, pp. 118–124.

[38] L. Yang and Y. J. S. Jiang, "A probabilistic preemptive scheme for providing service differentiation in OBS networks," in *Proceedings of IEEE GLOBECOM*, 2003.

[39] S. Lee, K. Sriram, H. Kim, and J. Song, "Contention-based Limited Deflection Routing in OBS Networks," in *Proceedings of IEEE Globecom*, 2003, pp. 2633-2637.

[40] S. Lee, L. Kim, J. Song, D. Griffith, and K. Sriram, "Dynamic Deflection Routing with Virtual Wavelength Assignment in Optical Burst-Switched Networks," *Photonic Network Communications,* vol. 9, pp. 347-356(10), May 2005.

[41] "The Network Simulator - ns-2," http://www.isi.edu/nsnam/ns/, February 2006.

[42] "OIRC OBS-ns Simulator," http://wine.icu.ac.kr/~obsns/index.php, February 2006.

[43] M. S. Taqqu, W. Willinger, and R. Sherman, "Proof of a Fundamental Result in Self-Similar Traffic Modeling," *ACMCCR: Computer Communication Review,* vol. 27, 1997.

Appendix

The following information may be found on the CD-ROM that includes the simulation software and the related materials:

- **Simulation Software**
- **Simulation manuals**
- **Research Articles and Related Papers**
- **Thesis Document and Thesis Drawings**

Before reading the code presented in this appendix, please refer to the OBS manuals (OBSManual.pdf and OIRC-OBS-manual.pdf) provided in the thesis CD-ROM.

The following code added in this section represents the important methods added by the author, which contributed to the implementation of this experiment.

SYSTEM CONFIGURATION

The methods in this section can be found in the **ns-obs-lib.tcl** file.

```
#method to set the optical link costs

Simulator instproc linkcost { node1 node2 cost } {


 $self instvar link_

    set id1 [$node1 id]
    set id2 [$node2 id]
if [info exists link_($id2:$id1)] {
        $self cost $node1 $node2 $cost
    }

}
```

```
#method to create several connections per edge pair and to specify the
traffic load in the network

Simulator instproc create-pareto-connection { nEdgenodes nConnections } {


        global par
        set count 1
        $self instvar Node_

      for { set j 0 }  { $j < $nEdgenodes } { incr j } {

              for { set i 0 } { $i < $nEdgenodes } { incr i } {

                      for {set k 0} {$k < $nConnections} {incr k} {
              if { $j != $i } {

              #puts "count = $count \t"
              incr count
              set parsource($j:$i:$k) [new Agent/UDP]
      #loss monitor instead of NULL for sink to measure number of bytes received
              set parsink($j:$i:$k) [new Agent/Null]


              $self attach-agent $Node_($j) $parsource($j:$i:$k)
              $self attach-agent $Node_($i) $parsink($j:$i:$k)
              $self connect $parsource($j:$i:$k) $parsink($j:$i:$k)


              set par($j:$i:$k) [new Application/Traffic/Pareto]
              $par($j:$i:$k) attach-agent $parsource($j:$i:$k)
              $par($j:$i:$k) set packetSize_ 1000
              $par($j:$i:$k) set burst_time_ 100ms
              $par($j:$i:$k) set idle_time_ 11ms
              $par($j:$i:$k) set rate_ 1Mb
              $par($j:$i:$k) set shape_ 1.2
                                                  }
                                              }
                                          }
```

```
}

#puts "created pareto connection between $src $des"

}
#method to start and stop the duration of the each simulation for pareto generated traffic

Simulator instproc obs-startup-pareto-application { starttime stoptime} {



        global par
        set u [ new RandomVariable/Uniform ]
        $u set min_ $starttime
        $u set max_ [expr $starttime + 0.001]

        foreach name [array names par] {
                $self at [$u value] "$par($name) start"
                $self at $stoptime "$par($name) stop"
                                }
puts "started pareto sources"
}
```

IMPLEMENTATION OF DEFLECTION ROUTING

The methods added in this section can be found in **ns-route.tcl, classifier-base.cc and route.cc.**

TCL function that returns the distance between source and destination (ns-route.tcl)

```
Simulator instproc hopcount {src dst} {



        $self instvar routingTable_ Node_ link_

        if ![info exists routingTable_] {
            puts "error: routing table is not computed yet!"
            return 0
```

```
        }

        set i $src
        set j $dst
        set distance 0
        set tmpfrom $i
        set tmpto $j

        while {$tmpfrom != $tmpto} {
                set tmpnext [$routingTable_ lookup \
                        $tmpfrom $tmpto]

                        set distance [expr $distance + \
                                [$link_($tmpfrom:$tmpnext) cost?]]
                        set tmpfrom $tmpnext
                        }

        return $distance
#return 0
        }
```

SEARCH method for alternate paths in static routing (route.cc).

```
int RouteLogic::search (char* asrc, char* adst,char * anhop, int& result)

{

#define ADJ(i, j) adj_[INDEX(i, j, size_)].cost
#define ADJ_ENTRY(i, j) adj_[INDEX(i, j, size_)].entry
#define ADJ_NEIGHBOUR(i, j) adj_[INDEX(i, j, size_)].neighbour   //pascal
#define ROUTE(i, j) route_[INDEX(i, j, size_)].next_hop
#define ROUTE_ENTRY(i, j) route_[INDEX(i, j, size_)].entry

        Tcl& tcl = Tcl::instance();
        int src = atoi(asrc) + 1;
        int dst = atoi(adst) + 1;
        int nhop  = atoi(anhop) + 1;
        int tmpdist  = 0;              //temporary distance
        int dist = INFINITY;
        int tmpalthop = INFINITY;         //temporary next hop
        int n = size_;
```

```
        int k = src;            //current source


        if (route_ == 0) {
                // routes are computed only after the simulator is running
                // ($ns run).
                tcl.result("routes not yet computed");
                return (TCL_ERROR);
        }
        if (src >= size_ || dst >= size_) {
                tcl.result("node out of range");
                return (TCL_ERROR);
        }


        for (int v = 1; v < n ; ++v)        {

//check to see if there is an adjacent different from next hop and itself
if (ADJ_NEIGHBOUR(k,v) != INFINITY && ADJ_NEIGHBOUR(k,v)!= nhop &&
ADJ_NEIGHBOUR(k,v) != k)
//  if (ADJ_NEIGHBOUR(k,v) != INFINITY && ADJ_NEIGHBOUR(k,v)!= nhop &&
ADJ_NEIGHBOUR(k,v) != k)

{

//      printf("src =  %d adj =  %d adj_neighbour =  %d \n", k-1 ,v-1 ,
ADJ_NEIGHBOUR(k,v)-1 );

for (int z = 1 ; z < n ; ++z) {

if (ADJ_NEIGHBOUR(v,z) != INFINITY && ADJ_NEIGHBOUR(v,z) != k ){
                tmpdist = getdistance(ADJ_NEIGHBOUR(v,z),dst);
ADJ_NEIGHBOUR(v,z), v ,dst, tmpdist);
                                if(tmpdist < dist) {
                                        dist = tmpdist;
                                        tmpalthop  = v;
                                                }
                                }
                        }
```

```
            }

        }
if(dist == INFINITY || tmpalthop == INFINITY){result = -1 ;}
else if(dist != INFINITY && tmpalthop != INFINITY) {result = tmpalthop-1 ; }
else { return (TCL_ERROR) ; }
return TCL_OK;
}
```

Method to handle the functioning of the control (or BHP) packet at every node along its path. Deflection routing is implemented in this method (classifier-base.cc)

```
void BaseClassifier::handleControlPacket( Packet *p ) {

int fdl_count; //GMG -- added local fdl_count variable for passing to schedule function
int alt_hop = -1;

    if( p == NULL )
        return;

if (type_==1){
  //  Debug::debug("control packet");
    Debug::markTr( address_, p );}

    hdr_cmn *ch = hdr_cmn::access( p );
    hdr_ip *iph = hdr_ip::access( p );
    hdr_IPKT *hdr = hdr_IPKT::access( p );

    // double check to see this packet is of type IPKT with prio = 1
    if( ( ch->ptype() != PT_IPKT ) ||  ( iph->prio_ != 1 ) ) {
        Debug::debug( "Error incorrect control packet type" );
        exit(1);
    }

//PASCAL ADDING TO MODIFY PACKET PARAMETERS
        //hdr->emission_prio_ =  hdr->emission_prio_++ ;
```

```
    //GMG -- for control packet, link receive method will not decrement
    //       edge node electronic buffer; therefore, set ebuf_ind to 0
    hdr->ebuf_ind = 0;

    // Retreive the lauc scheduler for the next hop for the dest addr**PASCAL**********
     LaucScheduler *ls = sg.search(getNextHop (iph->daddr()));


// cout <<"current address"<<address_<<"dest"<<iph->daddr()<<endl;

    if( ls == NULL ) {
        char debugStr[100];
        sprintf( debugStr, "Laucscheduler not found for destination %s at node %d",
          iph->daddr(), address_ );
        Debug::debug( __FILE__, __LINE__, debugStr );
        // if this error occur check the TCL initialization of the laucschedulers
        // also check if the nextHop method returns the current next-hop.
        exit (-1);
    }
    double bhpDur = ls->duration( ch->size() );
    double burstDur = ls->duration( hdr->C_burst_size() );
    double curTime = Scheduler::instance().clock();
    // Note: the bhp is scheduled to leave at bhpStartTime
    double bhpStartTime =curTime + proc_time_;
    // the earlies the burst is expected to arrive is after the offset time specified
    // anyway will be held until the bhp leave (input fdl) in that way the burst is
    // tried to synchronize to the offset time behind the burst
    //double burstStartTime  = bhpStartTime + BurstManager::offsettime() ;
    // Todo: Introduce the guard bands in now !



    //But note that hdr->offset_time_ is altered if FDLs are scheduled, at
    // all nodes; see below

    if (type_ == 1) //core node; note that for egress edge node,
                    //the current function is not invoked
      hdr->offset_time_ += (hdr->FDL_delay_ - proc_time_ -
                                    hdr->tx_delay_);
```

```
    //GMG -- added if block below to check if the newly adjusted offset time is negative or
zero.  If so, the
    //      BHP is dropped.  Note that the burst will have already been dropped, as it will
have already
    //      have arrived in this node.  Note also that the case of zero is degenerate; we
don't know if the
    //      burst event precedes or follows the BHP; we drop the BHP in this case as well.
    if( hdr->offset_time_ <= 0 )
    {
        //pascal checker to see why packets are dropped!!! eitha offset or no schedule
        counter++;
        printf ("offsetbased-drop count %d \n", counter );
       StatCollector &sc = StatCollector::instance();
       sc.updateEntry( "BHPDROP", sc.getValue( "BHPDROP" ) + 1.0 );
       drop(p);
       return;
    }

    double burstStartTime = curTime + hdr->offset_time_;

    fdl_count = hdr->fdl_count_; //GMG -- added passing of fdl_count to scheduler
    Schedule data = ls->schedData( burstStartTime, burstDur, fdl_count );



    //GMG -- Adjust hdr->offset_time_ for any FDLs that are scheduled
    if (FS_.option_ == 1) // max # FDLs used per node
      hdr->offset_time_ += (double)(fdl_count) * (FS_.fdl_delay_);
    else if (FS_.option_ == 2) // max # FDLs used per path
      hdr->offset_time_ += (double)(fdl_count - hdr->fdl_count_) *
                                        (FS_.fdl_delay_);

    hdr->fdl_count_ = fdl_count;
/****************START PASCAL SECTION*********************************/

if (data.channel() < 0){


//check the discard priority of the BHP/BURST
```

```
//if discard priority is greater than 0 then continue else get out and get on with life

        if (hdr->discard_prio_ > 0)
        {

//call the search function with current address and destination
// if search function returns a next alternative hop then attempt to schedule on that
channel
// if data channel is available, then reduce discard priority due to deflection and reroute
// else drop the packet of course

          int next_alt_hop = search_next_hop(iph->daddr(),getNextHop(iph->daddr()));
      //    ls = NULL; //setting back to null for future checks.

        if (next_alt_hop > -1) {
                ls = sg.search(next_alt_hop);

        if( ls == NULL ) {
         char debugStr[100];
          sprintf( debugStr, "Laucscheduler not found for destination %s at node %d",
         iph->daddr(), address_ );
          Debug::debug( __FILE__, __LINE__, debugStr );
        // if this error occur check the TCL initialization of the laucschedulers
        // also check if the nextHop method returns the current next-hop.
          exit (-1);
                }

    bhpDur = ls->duration( ch->size() );
    burstDur = ls->duration( hdr->C_burst_size() );


    // Note: the bhp is scheduled to leave at bhpStartTime
        //      printf ("old data channel %d \n", data.channel());
                data = ls->schedData( burstStartTime, burstDur, fdl_count );
        //      printf("attempt to reschedule for node %d \n" , next_alt_hop);
        //      printf("new data channel %d \n" .data.channel());
              alt_hop = next_alt_hop;
        //      printf("hop %d \n" ,alt_hop);
                }
```

```
        }
}
/*****************END PASCAL SECTION*****************************/
    if( data.channel() < 0 ) {


        char str[200];
      /*   sprintf( str, "Unable to schedule burst: %d in slot (%lf, %lf)",
           hdr->C_burst_id(), (burstStartTime * 1000.), (burstStartTime + burstDur) *
1000. );
         Debug::debug( str );
       printf ("dropped burst\n") ;*/


         StatCollector &sc = StatCollector::instance();
       //GMG - fixed following line; "BHPDROP" was written "BHPROP"
       sc.updateEntry( "BHPDROP", sc.getValue( "BHPDROP" ) + 1.0 );
       //Packet::free( p );
       //GMG -- changed the above from free to drop, to allow for trace objects
       drop(p);
       return;
    }

    //S 2004/08/18 Dr. Garner ------------------------------
    if (type_ == 0) //edge node; update offset to account for
                    //any electronic buffering of DB
      hdr->offset_time_ += (data.startTime() - burstStartTime);
    //E 2004/08/18 Dr. Garner ------------------------------

    Schedule control = ls->schedControl( bhpStartTime, bhpDur );

   if( control.channel() < 0 ) {
       char str[200];
      /*   sprintf( str, "Unable to schedule BHP for burst: %d in slot (%lf, %lf)",
           hdr->C_burst_id(), bhpStartTime * 1000., (bhpStartTime + bhpDur) * 1000. );
       Debug::debug( str ); */
```

```
        StatCollector &sc = StatCollector::instance();
        sc.updateEntry( "BHPDROP", sc.getValue( "BHPDROP" ) + 1.0 );
        //Packet::free( p );
        //GMG -- changed the above from free to drop, to allow for trace objects
        //GMG -- added restore of FDL scheduler state
        FS_.FdlSchedRestore();
        drop(p);
        return;
    }

        double stime = data.startTime();
        double etime = stime + burstDur;
        u_long chan = (u_long)data.channel();
   lswitch.add( (unsigned long)hdr->C_burst_id_, (u_long)0, (u_long)chan,
(double)stime, (double)etime, (double)etime ,(int) alt_hop );
       // lswitch.add( (unsigned long)hdr->C_burst_id_, 0, 0, 0.0, 0.0, 0.0 );

//        Scheduler::instance().schedule( slot_[iph->daddr()], p, burstStartTime -
curTime );
//GMG -- the BHP should be scheduled to leave the core or edge
//        classifier (i.e., leave the node) at the BHP start time;
//         not the burst start time
//       printf("ipd->daddress=  %d slot= %d and current= %d \n" , 2, slot_[2] ,
address_);

//printf("bhpid %d and alt_hop %d\n" , hdr->C_burst_id_ , alt_hop);

    //    Scheduler::instance().schedule( slot_[iph->daddr()], p, bhpStartTime - curTime );
//PASCAL CANCELLED SCHED

//pascal added scheduling to cater for possibility of alternate hop.

        if (alt_hop == -1) {
              Scheduler::instance().schedule( slot_[getNextHop(iph->daddr())], p,
bhpStartTime - curTime );
              }
        else  {
              /*PASCAL -  reduce discard priority and schedule for alternate hop
               -  add to number of deflections.-*/
```

```
            hdr->discard_prio_ = hdr->discard_prio_ - 1 ;
              hdr->deflections_ = hdr->deflections_ + 1;
              Scheduler::instance().schedule( slot_[alt_hop], p, bhpStartTime -
curTime );
          }
```

// Method to handle the functioning of a data-burst at every node along its path(classifier-base.cc)

```
void BaseClassifier::handleDataBurst( Packet *p ) {

   if( p == NULL )
      return;


   if (type_ == 1) {Debug::markTr( address_, p ); }

   hdr_cmn *ch = hdr_cmn::access( p );
   hdr_ip *iph = hdr_ip::access( p );
   hdr_IPKT *hdr = hdr_IPKT::access( p );

   if( ( ch->ptype() != PT_IPKT ) || ( iph->prio_ != 2 ) ) {
      Debug::debug( ___FILE___, ___LINE___,"Critical error: DataburstHandler received
a non-burst ipkt" );
      exit( -1 );
   }


      /**********************PASCAL HASH SECTION************************/

//pascal- lookup for alt hop before deletion in case there is one.
   HashEntry *he2 = lswitch.lookup ((unsigned long) hdr->C_burst_id() );
//      if(he2 != NULL)
//      {
 //       printf("burstid %d and althop%d \n", hdr->C_burst_id(),he2->alt_hop);
//      }
   HashEntry *he = lswitch.erase( (unsigned long) hdr->C_burst_id() );

   if( he == NULL ) {
      /* critical error bhp is ahead of the burst or no scheduler found*/
```

```
/* char str[100];
   sprintf( str, "Dropping burst: %d", hdr->C_burst_id() );
   Debug::debug( str ); */

StatCollector &sc = StatCollector::instance();
sc.updateEntry( "BURSTDROP", sc.getValue( "BURSTDROP" ) + 1.0 );


  hdr_cmn* tcpch;
int npkts = hdr->npkts();
Packet **tcp_pkt = (Packet**)p->accessdata();

while ( npkts--) {
    tcpch = hdr_cmn::access(*tcp_pkt);

     if ( tcpch->ptype() == PT_TCP)
    {
      sc = StatCollector::instance();
      sc.updateEntry( "TCPDROP", sc.getValue( "TCPDROP" ) + 1.0 );
    }
    else if (  tcpch->ptype() == PT_ACK)
    {
      sc = StatCollector::instance();
      sc.updateEntry( "ACKDROP", sc.getValue( "ACKDROP" ) + 1.0 );
    }
    else if (  tcpch->ptype() == PT_UDP)
    {
      sc = StatCollector::instance();
      sc.updateEntry( "UDPDROP", sc.getValue( "UDPDROP" ) + 1.0 );
    }
    else if (  tcpch->ptype() == PT_CBR)
    {
      sc = StatCollector::instance();
      sc.updateEntry( "CBRDROP", sc.getValue( "CBRDROP" ) + 1.0 );
    }
    else if (  tcpch->ptype() == PT_EXP)
    {
      sc = StatCollector::instance();
```

```
            sc.updateEntry( "EXPDROP", sc.getValue( "EXPDROP" ) + 1.0 );
          }
          else if (  tcpch->ptype() == PT_PARETO)
          {
            sc = StatCollector::instance();
            sc.updateEntry( "PARDROP", sc.getValue( "PARDROP" ) + 1.0 );
          }

          //free( *tcp_pkt );  //GMG -- fixed this; originally was tcp_pkt
                             //       rather than *tcp_pkt
          //GMG -- further change -- changed from free to drop to allow for trace objects
          drop( *tcp_pkt );
          tcp_pkt++;
        }
//Uncommented out block of code ends here
        //Packet::free( p );
        //GMG -- changed the above from free to drop, to allow for trace objects
        drop(p);
        return;
      } else {
        double curTime = Scheduler::instance().clock();
        double sendTime = 0.0;
        if( he->arrTime >= curTime ) {
          /* Life is simple the sending time is the arrTime - curtime */
          sendTime = he->arrTime - curTime;
        } else {
          /* Ok the burst came in late --*/
          sendTime = 0.0;
        }

// GMG the if block below is incorrect.  The DB should always be transmitted
// at sendTime.  We comment the block out, and use the statement in the
// else clause.
/*
        if( type_ == 0 ) {
          char str[100];
 */
          /* sprintf( str, "Node-type:%d Sending the burst:%ld at time: %lf, offsettime: %lf",
              type_, hdr->C_burst_id(),
```

```
            1000. * (Scheduler::instance().clock() + BurstManager::offsettime()),
            1000.* BurstManager::offsettime() ):
          Debug::debug( str ); */
/*
          Scheduler::instance().schedule( slot_[iph->daddr()], p,
BurstManager::offsettime() );
       }
       else
         Scheduler::instance().schedule( slot_[iph->daddr()], p, sendTime );
*/

  //     Scheduler::instance().schedule( slot_[iph->daddr()], p, sendTime );
 //pascal added hopcount
       hdr->hopcount_ = hdr->hopcount_ + 1;

//pascal added scheduling of next hop to include possible alternative hops
       if (hc2->alt_hop == -1){
      Scheduler::instance().schedule( slot_[getNextHop (iph->daddr())], p,
sendTime ); }//testing
       else{
       /*reduce discard priority before switching the burst
       burst would be dropped if less than 0 in any case since BHP would have been
dropped.
            - also add to number of deflections*/
       hdr->discard_prio_ = hdr->discard_prio_ - 1;
       hdr->deflections_ = hdr->deflections_ + 1 ;
      Scheduler::instance().schedule( slot_[hc2->alt_hop], p, sendTime ); }

      //GMG -- if this is an edge node, the first link recv method must
      //       decrement the electronic buffer fill.  Set ebuf_ind to 1
      //       and pointer to BaseClassifier for this case.  For core
      //       node, do nothing.
      if (type_ == 0)
      {
        hdr->ebuf_ind = 1;
        hdr->bc_ingress = this;
      }
   }
}
```

TOPOLOGY SETUP

This code shows the setup of the topology and the configuration parameters used for the simulation. They are located in the maincase.tcl file.

```
StatCollector set debug_ 0

Classifier/BaseClassifier/EdgeClassifier set type_ 0
Classifier/BaseClassifier/CoreClassifier set type_ 1
# Per node bhp processing time is 1 micro-second
source ../../lib/ns-obs-lib.tcl
source ../../lib/ns-obs-defaults.tcl
source ../../lib/ns-optic-link.tcl

set ns [new Simulator]
set nf [open p2p.nam w]
set sc [new StatCollector]
set tf [open trace01.tr w]
set ndf [open ndtrace01.tr w]

# dump all the traces out to the nam file
# note: as of now we do not support nam tracing
# @todo: intend to add nam-trace support
$ns namtrace-all $nf
$ns trace-all $tf
$ns nodetrace-all $ndf
#$ns rtproto Algorithmic

#=============================================================
====================#
# constant definitions
# set the offset time To to 10 - 20 microseconds
BurstManager offsettime 0.00005
#maxburstsize --change back to 1MB  = 1000000
BurstManager maxburstsize 1000000

#BurstManager bursttimeout 0.000008
BurstManager bursttimeout 0.2
#BurstManager burst_timeout 0.2
```

```
# set the bhp processing time 1 microsecond
Classifier/BaseClassifier/EdgeClassifier set bhpProcTime_ 0.000001
Classifier/BaseClassifier/CoreClassifier set bhpProcTime_ 0.000001

# total number of edge nodes
set edge_count 12
# total number of core routers
set core_count 6

# total bandwidth/channel (1mb = 1000000)  (1Gb  = 1000000000)
set bwpc 1000000000
#set bwpc
# delay in milliseconds
set delay 1ms
set delay2  2ms
set delay3  3ms
# total number of channels per link
set maxch 2
# number of control channels per link
set ncc 1
# number of data-channels
set ndc 1

# set the variables too.
$ns set bwpc_  $bwpc
$ns set maxch_ $maxch
$ns set ncc_  $ncc
$ns set ndc_ $ndc

#=============================================================
==================#
# support procedures

# finish procedure
proc finish {} {
    global ns nf sc tf ndf
    $ns flush-trace
    $ns flush-nodetrace
```

```
    close $nf
    close $tf
    close $ndf
    $sc display-sim-list

    puts "Simulation complete";
    exit 0
}

#create a edge-core-edge topology
Simulator instproc  create_topology { } {
    $self instvar Node_
    global E C
    global edge_count core_count
    global bwpc maxch ncc ndc delay delay2 delay3

    set i 0
    # set up the edge nodes
    while { $i < $edge_count } {
        set E($i) [$self create-edge-node $edge_count]
        set nid [$E($i) id]
       set string1 "E($i) node id:     $nid"
      puts $string1
        incr i
    }

    set i 0
    # set up the core nodes
    while { $i < $core_count } {
        set C($i) [$self create-core-node $core_count]
        set nid [$C($i) id]
       set string2 "C($i) node id:     $nid"
       puts $string2
        incr i
    }

        $self createDuplexFiberLink $E(0) $C(0) $bwpc $delay $ncc $ndc $maxch
        $self createDuplexFiberLink $E(1) $C(0) $bwpc $delay $ncc $ndc $maxch
        $self createDuplexFiberLink $E(2) $C(1) $bwpc $delay $ncc $ndc $maxch
```

```
        $self createDuplexFiberLink $E(3) $C(1) $bwpc $delay $ncc $ndc $maxch
        $self createDuplexFiberLink $E(4) $C(2) $bwpc $delay $ncc $ndc $maxch
        $self createDuplexFiberLink $E(5) $C(2) $bwpc $delay $ncc $ndc $maxch
        $self createDuplexFiberLink $E(6) $C(2) $bwpc $delay $ncc $ndc $maxch
        $self createDuplexFiberLink $E(7) $C(3) $bwpc $delay $ncc $ndc $maxch
        $self createDuplexFiberLink $E(8) $C(3) $bwpc $delay $ncc $ndc $maxch
        $self createDuplexFiberLink $E(9) $C(4) $bwpc $delay $ncc $ndc $maxch
        $self createDuplexFiberLink $E(10) $C(4) $bwpc $delay $ncc $ndc $maxch
        $self createDuplexFiberLink $E(11) $C(4) $bwpc $delay $ncc $ndc $maxch

        $self createDuplexFiberLink $C(1) $C(5) $bwpc $delay2 $ncc $ndc $maxch
        $self createDuplexFiberLink $C(3) $C(5) $bwpc $delay2 $ncc $ndc $maxch
        $self createDuplexFiberLink $C(4) $C(5) $bwpc $delay2 $ncc $ndc $maxch

       $self createDuplexFiberLink $C(0) $C(1) $bwpc $delay3 $ncc $ndc $maxch
        $self createDuplexFiberLink $C(1) $C(2) $bwpc $delay3 $ncc $ndc $maxch
        $self createDuplexFiberLink $C(2) $C(3) $bwpc $delay3 $ncc $ndc $maxch
        $self createDuplexFiberLink $C(3) $C(4) $bwpc $delay3 $ncc $ndc $maxch
        $self createDuplexFiberLink $C(4) $C(0) $bwpc $delay3 $ncc $ndc $maxch
        $self linkcost $C(0) $C(1) 3
        $self linkcost $C(1) $C(0) 3
        $self linkcost $C(1) $C(2) 3
        $self linkcost $C(2) $C(1) 3
        $self linkcost $C(2) $C(3) 3
        $self linkcost $C(3) $C(2) 3
        $self linkcost $C(3) $C(4) 3
        $self linkcost $C(4) $C(3) 3
        $self linkcost $C(4) $C(0) 3
        $self linkcost $C(0) $C(4) 3
        $self linkcost $C(1) $C(5) 2
        $self linkcost $C(5) $C(1) 2
        $self linkcost $C(3) $C(5) 2
        $self linkcost $C(5) $C(3) 2
        $self linkcost $C(4) $C(5) 2
        $self linkcost $C(5) $C(4) 2
    $self build-routing-table

}
$ns create_topology
```

```
#$ns create-pareto-connection $E(0) $E(2) 1
$ns create-pareto-connection $edge_count 84
set simstarttime 0.5
set simendtime 1.5
$ns obs-startup-pareto-application $simstarttime $simendtime
$ns at [expr $simendtime + 1.0] "finish"
$ns run
```